

AUGUST 1978 VOLUME 3, Number 8
$2.00 in USA
$2.40 in CANADA
BYTE
the small systems journal
ROBERT 78 TINNEY


see next page

In the Queue

BYTE August 1978
Volume 3 Number 8

Foreground

Background

Nucleus

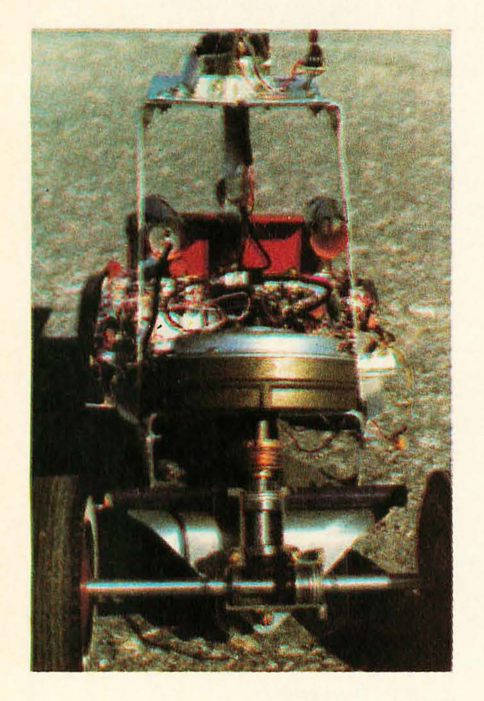


BYTE is published monthly by BYTE Publications Inc, 70 Main St, Peterborough NH 03458. Address all mail except subscriptions to above address: phone (603) 924-7217. Address all editorial correspondence to the editor at the above address. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of BYTE. Address all subscriptions, change of address, Form 3579, and fulfillment complaints to BYTE Subscriptions, PO Box 590, Martinsville NJ 08836. Second class postage paid at Peterborough NH 03458 and at additional mailing offices—USPS Publication No. 102410. Canadian second class registration No. 9321. Subscriptions are $15 for one year, $27 for two years, and $39 for three years in the USA and its possessions. In Canada and Mexico, $17.50 for one year, $32 for two years, and $46.50 for three years. $25 for a one year subscription by surface mail worldwide. Air delivery to selected areas at additional rates available upon request. $25 for a one year subscription by air delivery to Europe. Single copy price is $2.00 in the USA and its possessions, $2.40 in Canada and Mexico, $3.50 in Europe, and $4.00 elsewhere. Foreign subscriptions and sales should be remitted in United States funds. Printed in United States of America.

Subscription WATS Line: (800) 258-5485

In This BYTE

What are readers' experiences with building actual robotic mechanisms? Stephen A Allen and Anthony J Rossetti's commentary **On Building a Light-Seeking Robot Mechanism** describes their work in this area on an undergraduate engineering project. An on board computer helps their robot decide what action to take when avoiding obstacles between it and a light source.



page 24

Until now microprocessor users had the choice of using either an 8 bit or a 16 bit processor. With the advent of the 6809, Z-8000 and 8086, we now have available a hybrid machine with both 8 and 16 bit capabilities. This may have an effect on how language are written on the processors. Charles H Forsyth and Randall J Howard take a look at this in **Compilation and Pascal on the New Microprocessors.**

page 50

Do you need to perform extensive mathematical calculations, but fret over the time needed to write, debug and use floating point operations and transcendental functions? The new National Semiconductor MM57109 processor will help you with this problem. Turn to **The Number Crunching Processor** by Peter Nelson for details on how to interface an MM57109 with an 8080 system, and a review of this unique processor.

page 64

Pascal is the fastest growing new computer language. Few languages have the support from the typical university computer science department that Pascal has. To find out more about this fascinating language read Steven R Alpert's article, **Pascal, a Structurally Strong Language.**

page 78

A 179 year old android that can draw pictures and write poetry? It may sound like science fiction, but The Franklin Institute's Charles F Penniman reveals all in **Philadelphia's 179 Year Old Android.** *page 90*

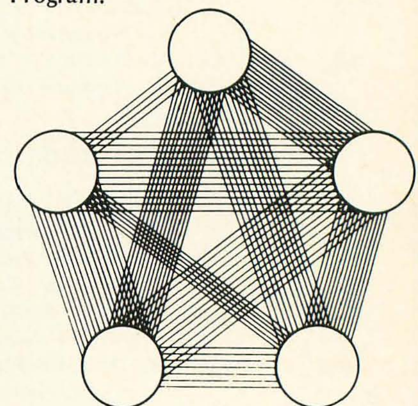
This month we continue Dr James Williams' 3 part series on **Antique Mechanical Computers** with **Part 2: 18th and 19th Century Mechanical Marvels.** Read about Jacquet-Droz's incredible writing and drawing automaton that contains the equivalent of three quarters of a million bits of read only memory and can draw intricate pictures and transcribe poetry.

page 96

Using a structured language such as Pascal helps the programmer easily organize the logic of a program. Often, it is not necessary to write a logic diagram for the program. To this end, David Mundie compares a well written Pascal program to a Warnier-Orr logic diagram in his article **In Praise of Pascal.** *page 110*

Business applications for personal computers are more and more in evidence these days. Pascal lends itself well to business applications with the addition of some special features described by Ken Bowles in **Pascal versus COBOL.** *page 122*

One of the newest developments in software is structured programming. Many features of the technique have been described, but often the actual procedure for constructing a structured program is not mentioned. Chip Weems describes the steps involved in **Designing a Structured Program.**



page 143

Light pens are one way to improve the user-computer interface, but there's an even more direct way: a noncontact scanning digitizer. Simply touch the screen of your video display to enter information! Steve Ciarcia shows you how in **Let Your Fingers Do the Talking: Add a Noncontact Touch Scanner to Your Video Display.**

page 156

If you like to gamble, but don't want to wait for legalized gambling in your state, try **JACPOT.** Author Edwin Hastings has written a straightforward BASIC simulation of a slot machine. Now you can gamble (for fun only, of course) to your heart's content without depleting your bank account. You can lose everything and then turn around and lose it again!

page 166

Pascal is an exciting language that can help you program more efficiently. It was developed in 1969 as an extension of the ALGOL family of languages. Author Allan Schwartz compares Pascal to BASIC, a language familiar to many BYTE readers, in **Pascal versus BASIC: An Exercise.**

page 168

Editorial

A Vision of an Industry

By Carl Helmers

In mid-March of this year, I finished a trip to the West Coast by having a day long meeting with Ken Bowles and his associates at the University of California, San Diego. The purpose of this meeting was to explore some of the possibilities which arise from the standardization of extensions to Niklaus Wirth's language Pascal, and the equally important implications of the technology of intermediate languages such as the optimized form of "P-code" developed at UCSD.

I came to this meeting with a background of familiarity with the reasons for encouraging highly structured languages such as Pascal. Before starting BYTE, I had been involved with the NASA HAL/S language developed by my employer of the time, Intermetrics Inc of Cambridge MA. I lived and breathed considerations of software reliability, ease of program design and the conceptual economy of a detailed program representation which doubles as the documentation of the algorithm. My personal experiences were with the context of the need to "man rate" the flight software of a contemporary spaceship through the use of high reliability software tools and techniques. These points are made elegantly in a number of books and papers which have been published on the subject to date.

What came out of this meeting with Ken Bowles is a vision of an important synthesis of machine independent software representations, the technology of printing machine readable software on paper, and the distribution of software in the form of conventionally printed and bound publications. It is a vision of what the software publishing business could look like over the course of the next few years.

Out of this vision of a machine independent software publishing industry comes a serendipitous justification for support of Ken Bowles' efforts to establish a "bandwagon" effect of support for the Pascal language and machine independent software systems. The purpose of this essay is to discuss the present dimensions of the software publishing problem, the technology which exists for preparing and printing machine readable representations, and the vision of machine independent software publishing which Ken Bowles and I saw inherent in the Pascal P-code technology as we discussed it that day.

Based on a computer graphic suggestion by students Joel McCormack and Owen Hampton at UCSD, we arranged with Russell Myers for this statement of an extreme opinion about Pascal. . . .



"UCSD Pascal"

Publishing Software

As the users of the personal computer expand in number, the means of distribution of software become critical to those who would distribute such software. In personal computing we are faced with a kind of problem which is completely new in the computer industries: the number of machines installed is becoming incredibly large by standards of the past 20 years, and the price paid per unit installation is becoming incredibly small. The computers which are a potential market for software are in the initial stages of becoming a mass market: too large a market for the custom craftsmanship of the traditional software vendor. To be convenient for the customers programs must be distributed with a machine readable copy which eliminates the need for hand key-

Continued on page 133



The answer is a

It's the serious solution to the small computer question.

Sol Systems are the key to effective, economical small computer power. Sol Systems give you the force of a powerful general purpose computer, the problem solving capability of high level languages and the operational simplicity of everyday office equipment.

From the ground up, Sol Systems were designed to do a complete job without adding a load of costly extras. In fact, when you compare the "everything included" price of a quality, field proven Sol System with anything else on the market, you'll be happily surprised to find out how little the extra performance and convenience costs.

For example, complete Sol Systems with 16,384 bytes of RAM memory start at less than $2500.* Expanded systems with 49,152 bytes of RAM memory, 1.5 million bytes of on-line disk memory, disk operating system and Extended Disk BASIC cost less than $8000.* Both systems are fully assembled, burned-in, tested and ready to go.

Sol Compatibility

Sol Systems feature the S-100 bus for pin-to-pin compatibility with a wide variety of add-on devices such as voice input and computer graphics. Standard Sol parallel and serial interfaces will drive most standard printers, modems and other peripherals.

A word about languages

No system is complete without software, and at Processor Technology we have tailored a group of high level languages, an assembler and other packages to suit the wide capabilities of our hardware.

Take a look at our exclusive Extended BASIC as an example. In cassette form, this BASIC features string and advanced

*U.S. prices only.



Sol System.

file handling, special screen commands, timed input, complete matrix, logarithmic and trigonometric functions, exponential numbers, 8 digit precision and square root. The language handles serial access files, provides tape rewind and offers cursor control for graphics capability.

The disk version has all the number crunching talents of the cassette BASIC plus instant access to data and programs on floppy disks. It includes random as well as sequential files and a unique ability to update sequential data in place.

Processor Technology FORTRAN is similar to FORTRAN IV and has a full set of extensions designed for the "stand alone" computer environment. Thousands of special application programs available through books and periodicals have already been written in this well established language.

Processor Technology PILOT is an excellent language for teachers. It is a string-oriented language designed expressly for interactive applications such as programmed instruction, drill and testing.

No wonder we call it the serious solution to the small computer question.

It's the small computer system to do the general ledger and the payroll. Solve engineering and scientific problems. Use it for word processing. Program it for computer aided instruction. Use it anywhere you want versatile computer power!

Sold and serviced only by the best dealers.

Sol Systems are sold and serviced by an outstanding group of conveniently located computer stores throughout the United States and Canada. They are also available in Australia, Europe, the United Kingdom, Central America, South America, Japan and Singapore.

For more information contact your nearest dealer listed on the following page. Or write Department B, Processor Technology Corporation, 7100 Johnson Industrial Drive, Pleasanton, CA 94566. Phone (415) 829-2600.

Circle 305 on inquiry card.

Processor Technology

Letters

HIDDEN GOLD IN THE TI-59?

In regard to "hidden gold" in the TI-59 (Webb Simmons' letter-interview, March 1978 BYTE, page 133) the only things I've come across after extensive investigation are the following:

1. According to the manual, the "decrement and skip on zero" instruction only references memories 00 to 09 (ie: the instruction may only be followed by a single digit for the memory reference). In fact, *any* available memory can be used, *except* for 40, which happens to be the numerical code for an indirect memory reference. You have to use the editing functions or some other trick to jam the 2 digit number in, but this isn't too much trouble; in the indirect mode, memory 40 may also be used, and there's no jamming problem at all.

2. This one is hardly hidden, but apparently (judging by the TI programs that came with the calculator) TI's designers didn't know about it: to make the display flash without altering anything else (including the contents of the display), the sequence "2nd operation 99" does very nicely; actually, "2nd operation anything larger than 39."

James G Owen
951 Dryden Rd
Ithaca NY 14850

TEXT EDITING

I sympathize and agree with your essay on the need for sophisiticated text editing (March 1978 BYTE, page 6).

I am a full-time writer who spends a great deal of time at my correcting Selectric II and who would appreciate technology which would eliminate crossing out and retyping my manuscripts as I unscramble the words that first poured out.

At some point, I send these sheets to a typist, and they come back finished. That, too, could be eliminated by a text editor controlling a fancy typewriter.

But I need even more the ability to write and edit on a video screen, then call for a final printout of the text on a

to store many standard paragraphs for compilation into letters and replies to advertisements.

Please keep me informed of your progress in your quest for quality editing at an affordable price.

Robert A Moskowitz
403 W School House Ln
Philadelphia PA 19144

A POINT OF INTEREST...

While waiting for somebody to build a system cheap and intelligent enough for me I read your paper with great interest. Amusingly, BYTE is the only English magazine I have seen which turns the text on the side of the cover upside down.

Mats Liljedahl
Kalenderv 31
415 11 SWEDEN

*And, you will notice, starting with the January 1978 BYTE, we did a 180° flip of the binding edge notation. Consistency was viewed as a virtue within the context of one year's worth of BYTE after we discovered the orientation of the binding text in the January 1977 issue!...*CH

DMA AND VIDEO: ARTICLES NEEDED

How about running a review or an article on direct memory access (DMA) in microprocessors? We video hacks would like to read how to sort out manufacturers' specs, how to work around their design "lemons," and how to make multichannel DMA work effectively towards fancy pictures.

Dr W R Levick
Dept of Physiology
John Curtin School of Medical Research
POB 334
Canberra City AUSTRALIA (ACT 2601)

A MORE INEXPENSIVE DIRECT VIDEO CONVERSION?

Microcomputer System Solution.

Announcing the IMSAI VDP-40.



You've decided you want a microcomputer DP center — but what to buy? A component system? A computer box here, a CRT box there, a keyboard box, a floppy disk box . . . A so called inexpensive $695 system? No disk; no way to add enough memory . . . and if you could, it's not inexpensive anymore; and you still wind up with a collection of boxes.

The IMSAI VDP-40 is a fully integrated Video Data Processing system and comes equipped with your choice of 32K or 64K RAM on our own advanced RAM III board. In one cabinet, the VDP-40 combines a professional keyboard, heavy-duty power supply, twin mini-floppies, a multi-slotted motherboard, 9-inch CRT, plus RAM at a new price/performance never before achieved.

System Expansion? Extra slots in our S-100 bus motherboard and our new power supply allow almost unlimited expansion.
Need more disk expansion? A controller option of the VDP-40 allows expansion to greater than 4.5M Bytes.
Add a line printer, an IBM compatible tape drive, a modem: all are available, with the interfaces and software to make it work for you.

IMSAI's VDP-40 price/performance means every home or business can now afford a complete DP center. Check us out. IMSAI has what you want and what you need. Visit your dealer or write us directly.
Features subject to change without notice.

IMSAI®

The Standard of Excellence in Microcomputer Systems

IMSAI Manufacturing Corporation,
Dept. EWN, 14860 Wicks Boulevard
San Leandro, CA 94577
(415) 483-2093 TWX 910-366-7287

IMSAI EUROPE
Europa Terrassen
8 Rue Jean Engling
Dommeldange, Luxembourg
43-67-46 Telex: 1428

Price/Performance no one else has put together.

Circle 175 on inquiry card.



A Letter Exchange: Extending S-100 Bus?

When John McCallum in the March 1978

output port, a D/A converter puts its output

Why Apple II is the world's best selling personal computer.

Which personal computer will be most enjoyable and rewarding for you? Since we delivered our first Apple® II in April, 1977, more people have chosen our computer than all other personal computers combined. Here are the reasons Apple has become such an overwhelming favorite.

Apple is a fully tested and assembled mainframe computer. You won't need to spend weeks and months in assembly. Just take an Apple home, plug it in, hook up your color TV* and any cassette tape deck—and the fun begins.

To ensure that the fun never stops, and to keep Apple working hard, we've spent the last year expanding the Apple system. There are new peripherals, new software, and the Apple II Basic Programming Manual. And wait till you see the Apple magazine to keep owners on top of what's new.

Apple is so powerful and easy to use that you'll find dozens of applications. There are Apples in major universities, helping teach computer skills. There are Apples in the office, where they're being programmed to control inventories, chart stocks and balance the books. And there are Apples at home, where they can help manage the family budget, control your home's environment, teach arithmetic and foreign languages and, of course, enable you to create hundreds of sound and action video games.

When you buy an Apple II you're investing in the leading edge of technology. Apple was the first computer to come with BASIC in ROM, for example. And the first computer with up to 48K bytes RAM on one board, using advanced, high density 16K devices. We're working to keep Apple the most up-to-date personal computer money can buy. Apple II delivers the features you need to enjoy the real satisfaction a personal computer can bring, today and in the future.

15 colors & hi-resolution graphics, too.

Don't settle for a black and white display! Connect your Apple to a color TV and BASIC gives you instant command of three display modes: Text, 40h x 48v Color-graphics in 15 colors, and a 280h x 192v High Resolution array that lets you plot graphs and compose 3-D images. Apple gives you the added capability of combining text and graphics, too.

Back to basics, and assembly language too.

Apple speaks three languages: fast integer BASIC, floating point BASIC for scientific and financial applications, and 6502 assembly language. That's maximum programming flexibility. And, to preserve user's space, both integer BASIC and monitor are permanently stored in 8K bytes of ROM, so you have an easy-to-use, universal language instantly available. BASIC gives you graphic commands: COLOR=, VLIN, HLIN, PLOT and SCRN. And direct memory access, with PEEK, POKE and CALL commands.

Software: Ours and yours.

There's a growing selection of pre-programmed software from the Apple Software Bank—Basic Finance, Checkbook, High Resolution Graphics and more. Now there's a User Section in our bank, to make it easy for you to obtain programs developed

by other Apple owners. Our Software Bank is your link to Apple owners all over the world.

Alive with the sound of music.

Apple's exclusive built-in speaker delivers the added dimension of sound to your programs. Sound to compose electronic music. Sound to liven up games and educational programs. Sound, so that any program can "talk" back to you. That's an example of Apple's "people compatible" design. Another is its light, durable injection-molded case, so you can take Apple with you. And the professional quality, typewriter-style keyboard has n-key rollover, for fast, error-free operator interaction.

Apple is the proven computer.

Apple is a state-of-the-art single board computer, with advanced LSI design to keep component count to a minimum. That makes it more reliable. If glitches do occur, the fully socketed board and built-in diagnostics simplify troubleshooting. In fact, on our assembly line, we use Apples to test new Apples.

*Apple II plugs into any standard TV using an inexpensive modulator (not included).

**In California, call 408/996-1010.

Apple peripherals are smart peripherals.

Watch the far right column of this ad each month for the latest in our growing family of peripherals. We call them "intelligent interfaces." They're smart peripherals, so you can plug them in and run them from BASIC without having to develop custom software. No other personal computer comes close to Apple's expandability. In addition to the built-in video interface, cassette I/O, and four A/D inputs with two continuously variable game paddles, Apple has eight peripheral slots, three TTL inputs and four TTL outputs. Plus a powerful, state-of-the-art switching power supply that can drive all your Apple peripherals.

Available now.

Apple is in stock and ready for delivery at a store near you. Call us for the dealer nearest you. Or, for more details and a copy of our "Consumer's Guide to Personal Computers," call 800/538-9696** or write Apple Computer, Inc., 10260 Bandley Drive, Cupertino, CA 95014.



Circle 14 on inquiry card.



New from Apple.

Introducing Disk II™: instant access to your files.

Our newest peripheral is Disk II, a high-density 5¼" floppy disk drive for fast, lowcost data retrieval. It's perfect for storing large bodies of data such as household finances, address files and inventories; you can find any record in just half a second. No more searching through stacks of cassettes; with a few keystrokes, your system will load, store and run any file by name.

Disk II consists of an intelligent interface card, a powerful Disk Operating System (DOS), and one or two minifloppy drives. Your Apple will handle up to seven interface cards and fourteen drives, for control of nearly 1.6 megabytes of data, with no expansion chassis. The combination of ROM-based bootstrap loader and an operating system in RAM provides complete disk handling capability, including these special features:
• Soft sectored • Random or sequential file access • Program chaining capability • Universal DOS command processor works with existing languages and monitor • Full disk capability in systems with as little as 16K RAM • Storage capacity: 113 kilobytes/diskette.

See Disk II now at your Apple dealer. Sold complete with controller and DOS at $495.†

Peripherals in stock

Hobby Board (A2B0001X), Parallel Printer Interface (A2B0002X), Communication Interface (A2B0003X), Disk II (A2M0004X).

Coming soon

High speed Serial Interface, Printer II, Printer IIA, Monitor II, Modem IIA.

†Price subject to change without notice.

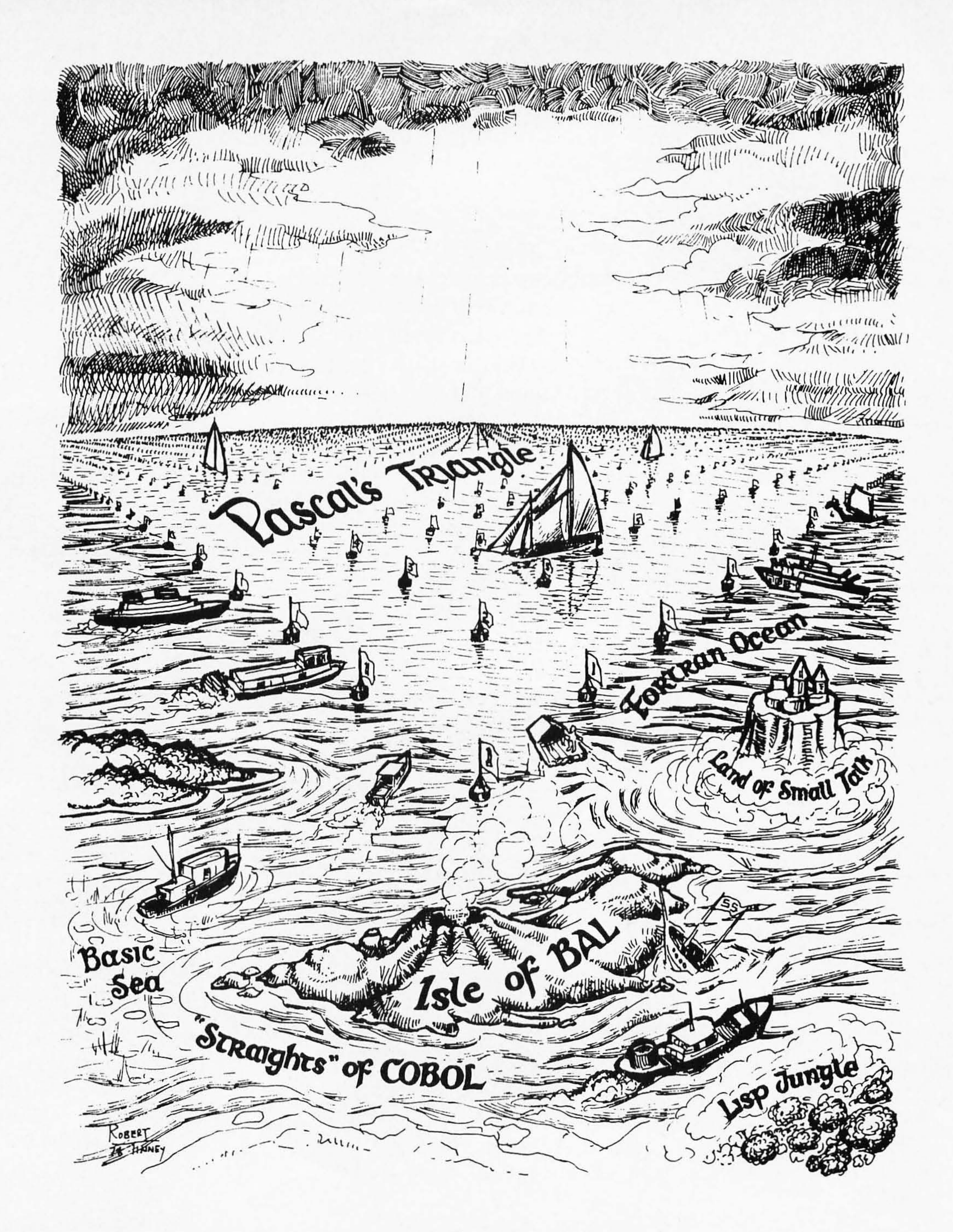
Circle 15 on inquiry card.

Programming is a snap! I'm halfway through Apple's BASIC manual and already I've programmed my own space wars game.

Those math programs I wrote last week–I just rewrote them using Apple's mini-assembler and got them to run a hundred times faster.

Apple's smart peripherals make expansion easy. Just plug 'em in and they're ready to run. I've already added two disks, a printer and the communications card.





About the Cover

by Carl Helmers

It is rare when one can indulge in one's prejudices with relative impunity, poking a bit of good humored fun to make a point. The design of the cover, entitled "Pascal's Triangle" provided just such an opportunity. The cover was executed by Robert Tinney, but the prejudices are all mine and were given to him as a fairly detailed script. The point is that Pascal is here, it is consistent with use by small computers, such as many readers own, and it is available in the form of the UCSD software system at quite a nominal charge above the cost of the hardware required. While today it requires a computer at the high end of the personal computer range of pricing, the utility of the language and advances in both magnetic media and read only memory technology should lower the price of the minimum hardware requirements considerably over the course of the next year or two. With that point, we present "Pascal's Triangle."

The primary allegory of the cover is of course the inversion of the "Bermuda Triangle" myth's theme to show smooth waters. The triangle is an unbounded tri-

Now there are at least 102K more reasons to buy the Heathkit H8 Personal Computer–

The WH17 Floppy!



If you read this magazine, you've probably seen our ads on the H8. It's a neat, well-designed machine with a lot of fine features including an intelligent front panel, octal keyboard entry and display, plenty of room for memory and I/O expansion, and lots of other practical "goodies," including the popular 8080A CPU. H8 owners have praised it for its convenience, versatility, ease of operating and general performance characteristics. In fact, if the H8 was "missing" anything, it was a floppy disk storage system.

Well, not any more! The WH17 Floppy is now a reality and it brings extraordinary power and versatility to your H8 computer system. Programs are loaded in seconds instead of minutes so you're up and running fast; you can store literally hundreds of programs on a single disk so you have lots of storage space and immediate access to all your programs; you can update files instantly and efficiently to save time. The WH17 is far more versatile and reliable than paper tape or cassette storage systems. It truly makes the H8 a "total capability" computer system!

WANGCO 82 Disk Drive Unit

Storage media for the WH17 is the standard hard-sectored 40-track diskette. Measuring just 5.25" in diameter, each disk offers access to better than 102K bytes of available program and data storage area. The drive system used in the WH17 is the famous WANGCO model 82, a performance-proven drive providing accurate high-speed data access. Specifications of this drive include a conservative 30 mS track-seek time and typical random sector access times of less than 250 mS. Compare. These figures are considerably better than you'll find for most equivalent competitive drives.

A diskette containing all operating systems software for the WH17 is sold separately for $100. This software includes the Heath Disk Operating System (HDOS) with its unique diagnostic for floppy evaluation and optimization; the BUG-8 console debugger; TED-8 text editor; HASL-8 assembly language and extended Benton Harbor BASIC with files. An optional plug-in second drive is also available for $295.

The WH17 Floppy Disk system is now available fully assembled and tested. With single drive and controller for only $675. A kit version will be available later. The H8 requires 16K of RAM to use the WH17 and its operating system to their fullest capabilities.

Prices are mail-order net FOB, Benton Harbor, Michigan. Prices and specifications subject to change without notice.

HEATHKIT COMPUTERS

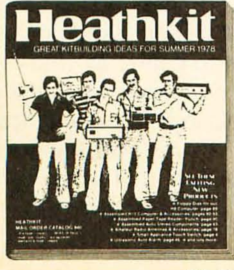
System Engineered for Personal Computing

WH17



FREE!

Be sure to fill out postcard to order your FREE catalog. If card is missing, write Heath Company, Dept. 334-440, Benton Harbor, Michigan 49022.



CP-152
Circle 160 on inquiry card.

angular array of buoys numbered with the binomial expansion coefficients. These coefficients are "Pascal's Triangle" as any high school algebra student will have learned. Pascal's Triangle on the cover is of course embedded in a matrix of the waters of the FORTRAN Ocean of computer languages, named after the pioneering widely used high level language FORTRAN, and its descendents FORTRAN II, FORTRAN IV, WATFOR, WATFIV and even the mildly reformed RATFOR.

A prominent island graces the bottom of the picture, the Isle of BAL with its rocky and desolate surface. Note the great JCL barrier reef which surrounds the Isle of Bal and borders the Straights of COBOL along the bottom edge of the picture. These dangerous and unstructured reefs have sunk more than one ship on their treacherous shoals, including the good ship SS OS of all encompassing (360 degree) fame. (Much commercial traffic is seen in the sea lanes of the Straights of COBOL.)

In the lower right part of the picture where the Straights of COBOL meet the main body of the FORTRAN Ocean, we see a curious fog bank (imagining a view from the deck of a ship in the waters). It is said that this fog bank is always present, hiding the exotic and mysterious jungles of LISP. While unseen by normal mortals, our gods' eye view of the picture shows the brilliant tropical algorithms, the fabulous nodes growing on trees like in some Eldorado of programming. But who can see brilliance through a fog bank?

Travelling upward (in the picture) through heavy seas we come to the pinnacle, a snow white island rising like an ivory tower out of the surrounding shark infested waters. Here we find the fantastic kingdom of small talk, where great and magical things happen. But alas, just as the impenetrable fog bank around the jungles of LISP hide it from our view, the craggy aloofness of the kingdom of small talk keeps it out of the mainstream of things.

Turning our attention to the lower left part of the picture, we see the famous Floating Point separating the FORTRAN Ocean mainstream from the interactive and weed filled Sea of BASIC.

To all the relative disorder and chaos of the waters of the FORTRAN Ocean and its adjacent coastal features, the smooth, calm infinity of Pascal's Triangle provides a brilliant contrast. We note vessels ranging from the commercial freighters to pleasure boats to the rafts of hobbyists to the military fighting ships heading for the calm waters of Pascal's Triangle.

To complete the mythology, we find within Pascal's Triangle numerous examples of ships enjoying the smooth sailing and untroubled waters.

Is this an adequate picture? Computer languages are like philosophies in many respects, which is to say that the reasons for an enthusiasm are often hard to attribute to anything other than aesthetic grounds. But as in philosophies and religions, conversions do occur from time to time. Very often in today's microcomputer world, we find the case of the engineer or systems programmer who has been using an assembler (if anything at all) as the first and only software development tool. Such a person will often discover BASIC, FORTRAN, APL, COBOL (yes, even people with engineering backgrounds sometimes see COBOL as a first high level language) or language X. When language X is discovered, the advantages of the high level language *technique* often become confused with the specific example—and the enthusiasm which comes with the powerful elixir of automated programming aids turns that person into an X language convert with an almost religious fervor.

As the new convert proceeds to use the language, he or she also discovers its inadequacies in detail errors. And the X language devotee starts inventing this or that perfect extension, a new superset of X, which is endowed with even better properties. This particular inventiveness syndrome is most pronounced in compiler implementors since they are in a position to "do something about" the older language by ad hoc implementing personally meaningful extensions when putting a new compiler up.

What has resulted, viewing from the big picture, is a range of languages, each reflecting the context of the group of implementors who are responsible for its creation. Pascal in this global context must be viewed as but another step in that natural sequence of human events.

I personally like Pascal as a method of expressing programs, because of a number of arguments supported by my own prior experience using languages including macroassemblers, BASIC, FORTRAN, PL/I, HAL/S, JOVIAL, XPL and a bit of PL/M.

As a potential user, try a few programs, see if you like the style of expression involved, and if the price is right, that may be the system for you. If you like the arguments presented for Pascal in this issue and by examples in issues to come, by all means express your interest to manufacturers. This issue is a conscious attempt to communicate some of the flavor of Pascal with a spirit of fun and an understanding that even Pascal may not be the be all and end all of computer languages.■



If you are interested in an ultra high performance personal computer which can be fully expanded to a mainframe class microcomputer system, consider the C2-8P.

Features:

- Minimally equipped with 8K BASIC-in-ROM, 4K RAM, machine code monitor, video display interface, cassette interface and keyboard with upper and lower case characters. (Video monitor and cassette recorder optional extras.)
- The fastest full feature BASIC in the microcomputer industry.
- Boasts the most sophisticated video display in personal computing with 32 rows by 64 columns of upper case, lower case, graphics and gaming elements for an effective screen resolution of 256 by 512 elements.
- The CPU's direct screen access, coupled with its ultra fast BASIC and high resolution, makes the C2-8P capable of spectacular video animation directly in BASIC.
- Fully assembled and tested: 8 slot mainframe class microcomputer, six open slots for expansion. Supports Ohio Scientific's ultra low cost dynamic RAM boards or ultra high reliability static RAMs.
- The C2-8P can support more in-case expansion than its four nearest competitors combined.
- The C2-8P is the only BASIC-in-ROM computer that can be directly expanded today to a complete business system with line printer and 8″ floppy disk drives.
- It is the only personal class computer that can be expanded to support a Hard Disk! (CD-74)

The C2-8P is the fastest in BASIC, has the most sophisticated video display and is the most internally expandable personal computer. Therefore, it should be the highest priced?

Wrong: The C2-8P is priced considerably below several models advertised in this magazine. The C2-8P is just one of several models of personal computers by Ohio Scientific, the company that first offered full feature BASIC-in-ROM personal computers.

For more information, contact your local Ohio Scientific dealer or the factory at (216) 562-3101.

OHIO SCIENTIFIC

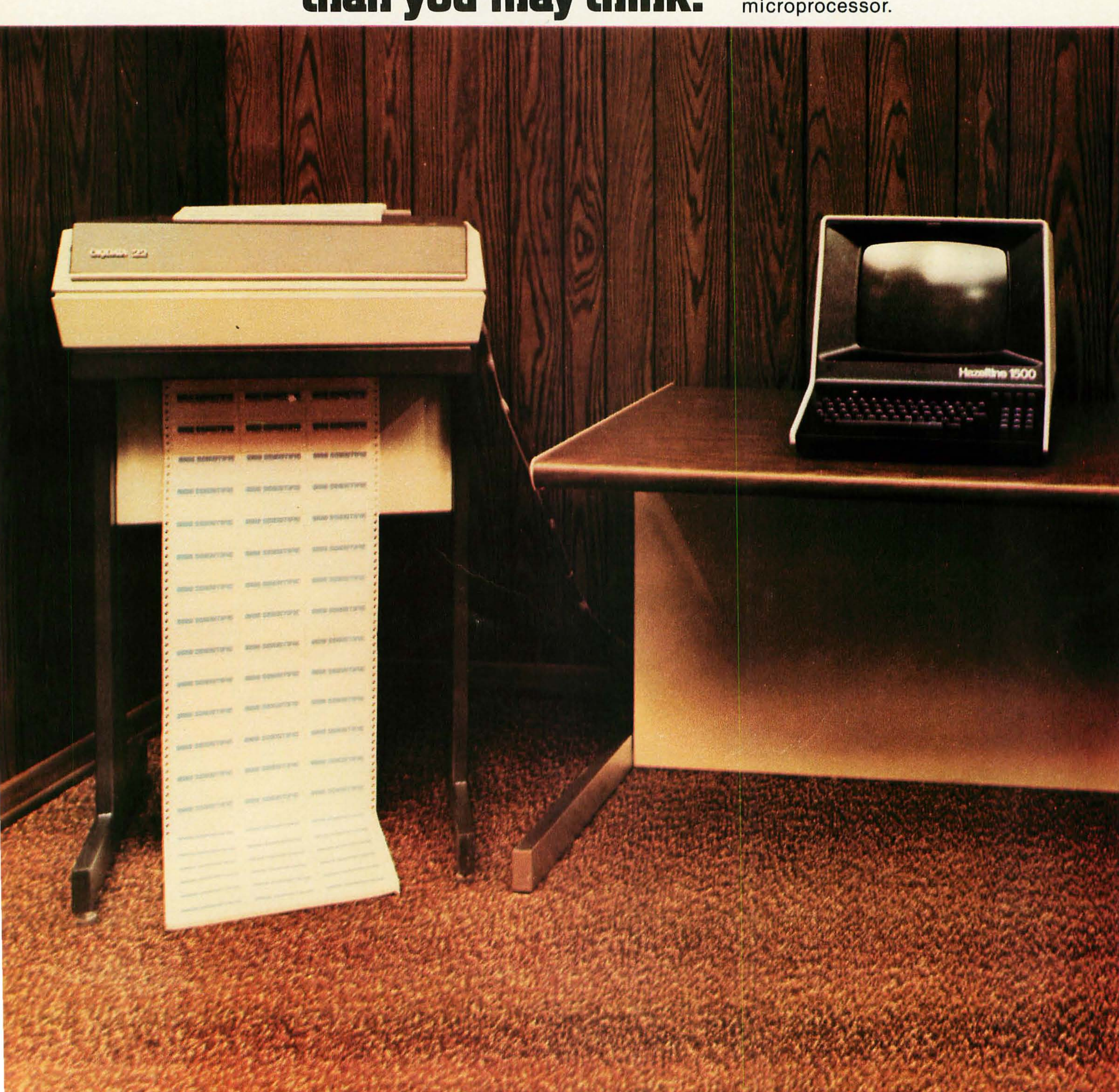
1333 S. Chillicothe Road • Aurora, Ohio 44202

The C3-B
by Ohio Scientific

The world's most powerful microcomputer system is far more affordable than you may think.*

STANDARD FEATURES:

- 74 million byte Winchester technology disk drive yields mainframe class file access speeds and capacity.
- High level data file software makes high performance file structures like multikey ISAM easy to use.
- Triple processor CPU with 6502A, 6800 and Z-80 gives the programmer the best of all worlds in performance and versatility.
- Switchable and programmable CPU clocks at 1, 2 and 4 MHz yield maximum performance from each microprocessor.



On Building a Light-Seeking Robot Mechanism

Stephen A Allen
POB 2281
Leucadia CA 92024

Tony Rossetti
1455 Arbor Av
Los Altos CA 94022

The idea of the Tee Toddler was born during the summer of 1976. We, as two undergraduate engineering students at Rice University in Houston, wanted to design a system using as much applicable electrical engineering as possible which could act on its own intelligence and which could also learn from its mistakes. We wanted to incorporate state of the art electronics and actually develop a piece of working hardware. As a three credit hour course for two semesters we designed and built a small "robot" car, the Tee Toddler.

The car is designed to track toward a shining light. It accomplishes this with the help of two processors: an on board Z-80 microprocessor which communicates with a PDP-11 minicomputer over a two way digital radio link. The source light the Tee Toddler searches for can be anywhere on the horizon. This light is detected by a rotating eye which scans a 360 degree view five times per second (see photo 1). There is also an ultrasonic sonar system capable of scanning simultaneously to the left and right of center to detect objects in the car's forward path (see photo 2). It can give ranges of up to five feet with 9 inch accuracy. The car has three forward and reverse speeds, five steering positions and a turning radius of five feet. Other standard equipment includes a front contact sensing bumper to detect objects which the sonar missed; a source light monitor to determine if the car is at its destination; a source light verifier to indicate whether the car has gone behind something which blocks the source light; whitewall tires and positive traction rear end. Most power requirements were met by regulating a 12 V rechargeable battery. The Tee Toddler is a closed loop system. It constantly updates its knowledge of where the light is and what obstacles are in the way; thus operating as a self-sufficient real time system. Great flexibility inherent in the two processor system allows for development of the car's intelligence. The programmer has lots of freedom in deciding how the car should deal with differing situations. This freedom in configuring the system between the computers and a moving object is the true beauty of the Tee Toddler. The duties of each processor are different. The on board Z-80 handles the car's reflex maneuvers; the PDP-11 makes both real time navigational decisions and can also generate a better path for the car to take on a second trip over the same obstacle course toward the light. The normal mode of operation is set up with the PDP-11 in control of the car via the radio. The Z-80 is operated in an

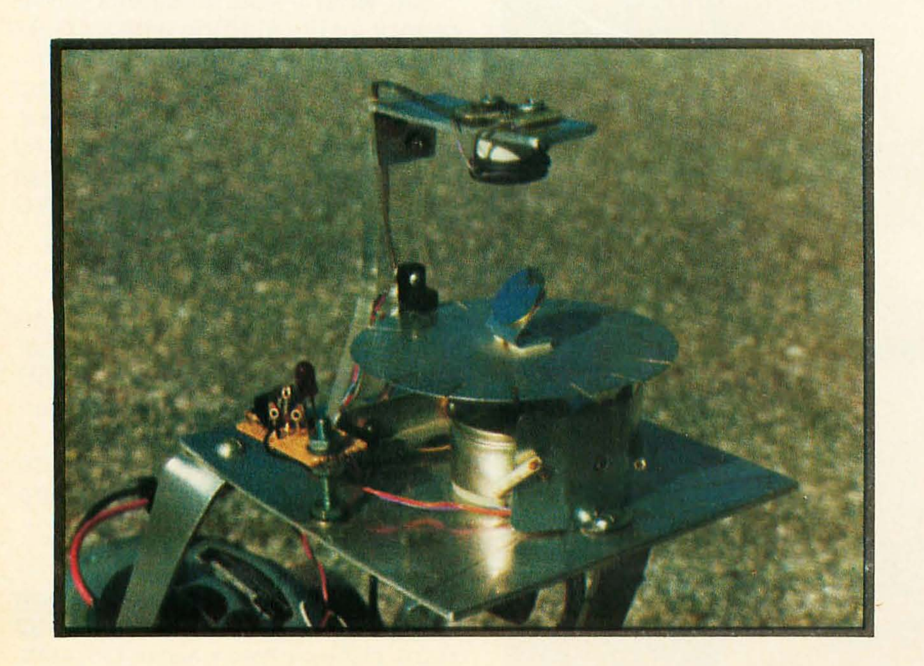


Photo 1: The primary sensor of Tee Toddler is this photoelectric horizon scanner. The "eye" mounted on a Plexiglas standard and metal bracket scans a 360° field. The flat mirror rotates at five revolutions per second deflecting light into the phototransistor eye through a 45° angle. The position of the mirror is resolved into one of 16 angular states by a slotted disk which passes through an optocoupler.

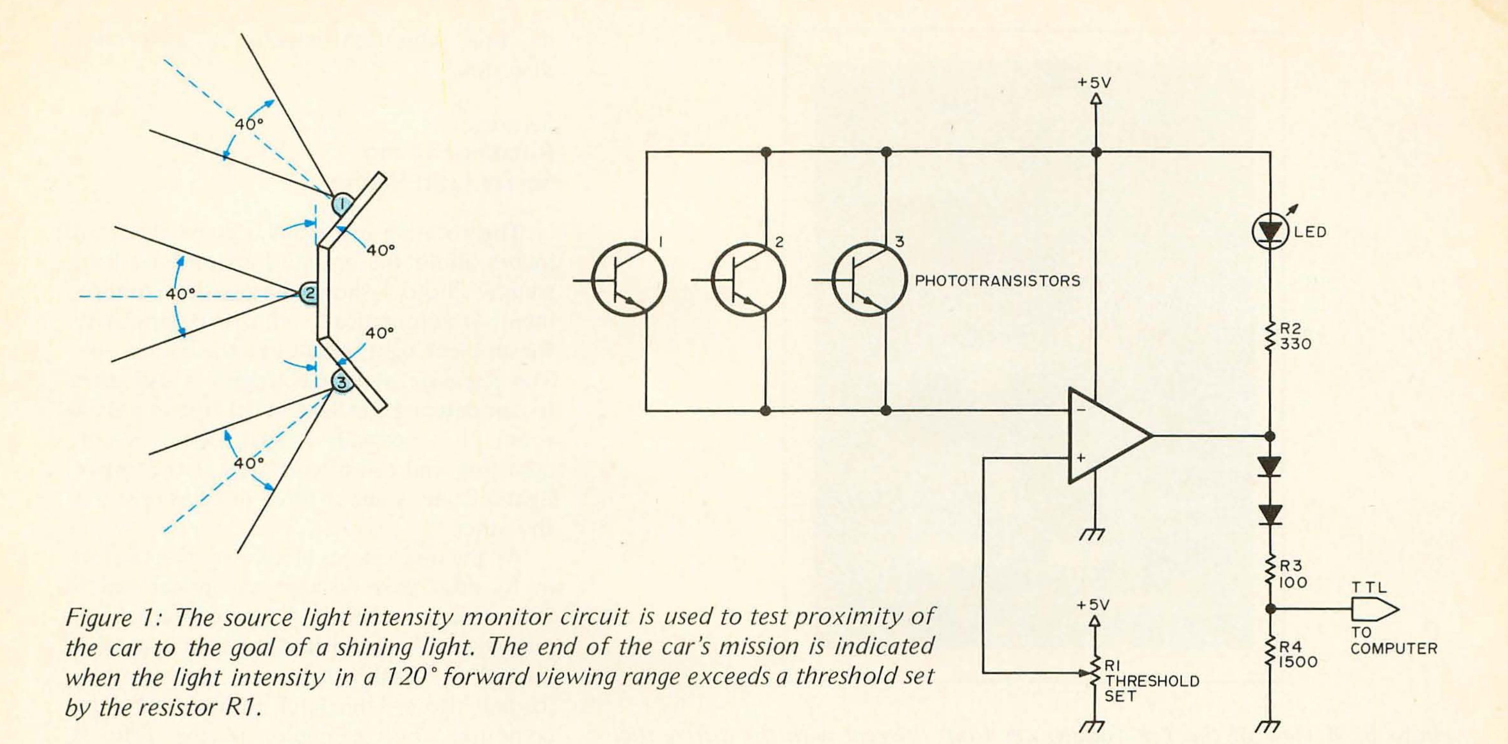


Figure 1: The source light intensity monitor circuit is used to test proximity of the car to the goal of a shining light. The end of the car's mission is indicated when the light intensity in a 120° forward viewing range exceeds a threshold set by the resistor R1.

interrupt mode. It stands by and records all course changes the car makes each time an obstacle is detected by sonar and also records each navigational correction made enroute to the light. In the case of collision or loss of the light, the Z-80 takes over control and remedies the situation before it returns control of the car to the PDP-11. At the end of the trip to the light the Z-80 dumps all the course change vectors it has recorded (steering setting and distance traveled) to the PDP-11 over the radio channel. The PDP-11 then determines a better path for the car to take over the same course on a second run.

Sensors: the Bumpers

The bumper is needed only in case an object is encountered in front of the car which is too narrow to be seen by the sonar. The bumper has two microswitches behind it, one on each side. It pivots in the center so that a contact on only one side will depress only one switch; however if the collision is head-on, both switches will be depressed. Two bits, one for each switch, are sent to the Z-80 computer.

Sensors: Source Light Intensity Monitor

The source light is the car's destination. The intensity monitor consists of three phototransistors which sense the intensity of light until the car is close enough to the source light (a foot) to stop; mission accomplished (see figure 1). An angle of 120 degrees is monitored, so the car must make

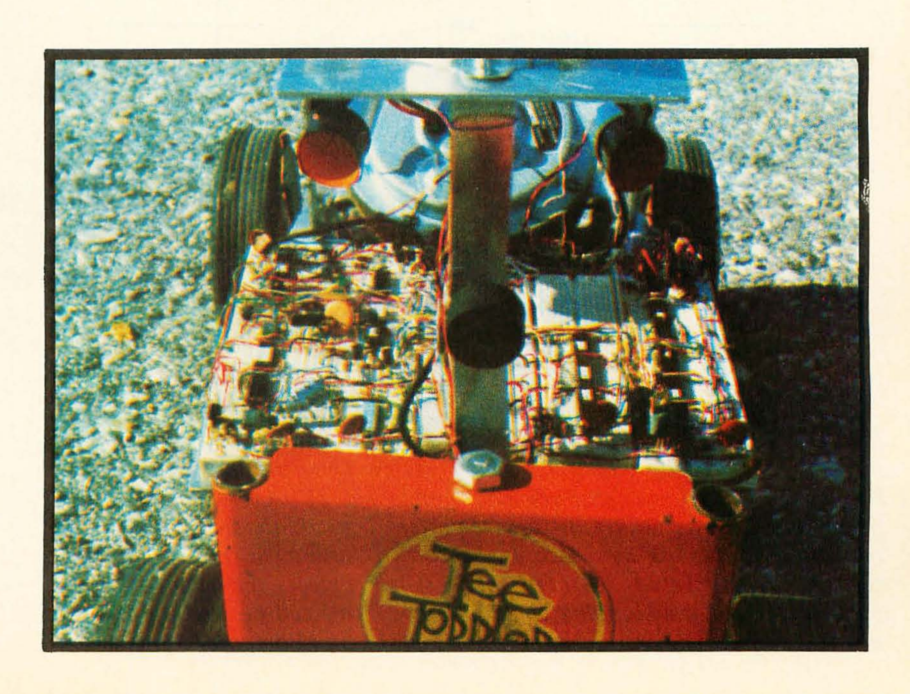


Photo 2: Tee Toddler's sonar system transducers are illustrated in this front view. The black object in the center of the picture is the sonar transmitter which emits periodic pulses of sound at 40 kHz. The two cup-like objects with red interiors (equally spaced to the left and right of the center of the picture toward the top) are the receiving microphones. The sonar drive electronics of this system resolves four distance states on each receiving microphone with a maximum range of about five feet and an accuracy of about nine inches.

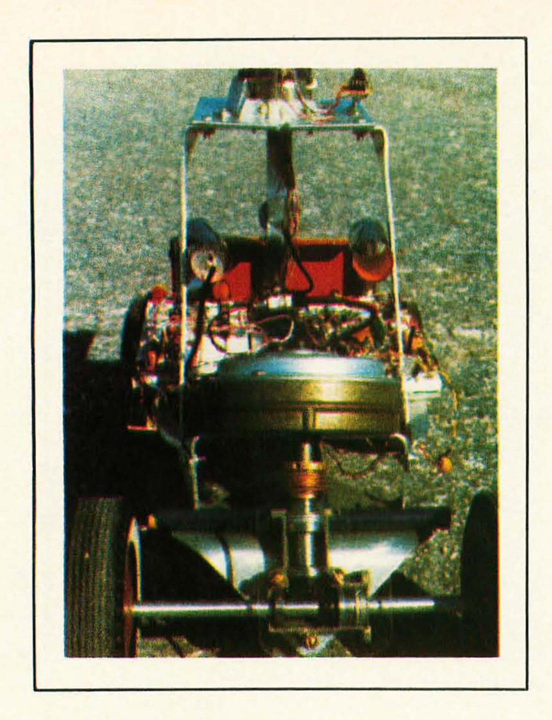


Photo 3: A view of the Tee Toddler car from the rear with the differential and drive motor visible. (The battery and rear deck have been removed for purposes of this photograph.)

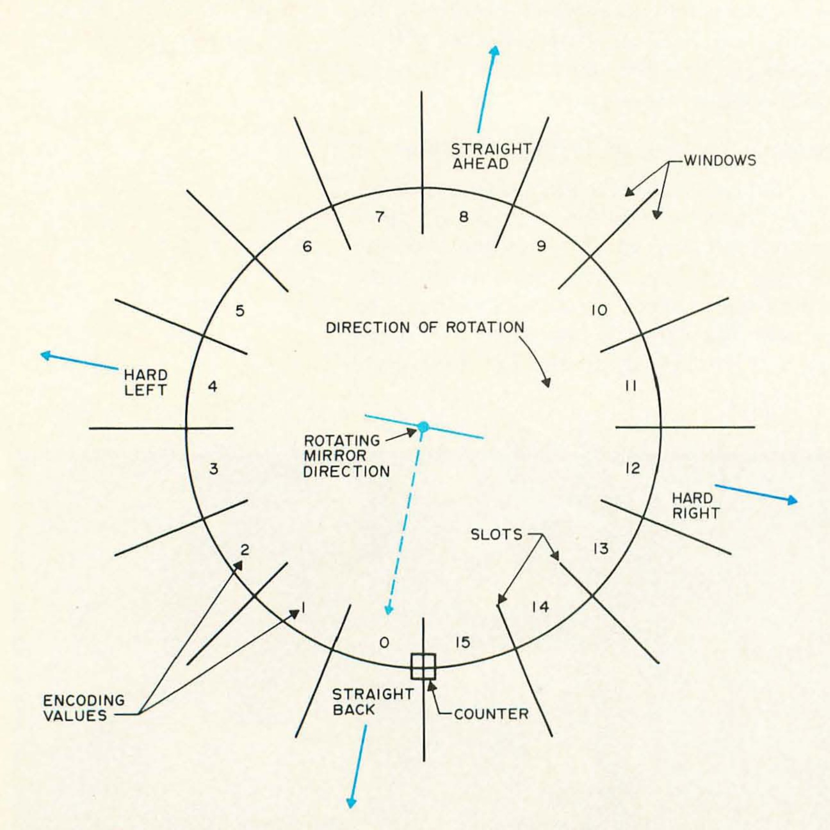


Figure 2: The direction of the light source relative to the forward direction of travel is measured by one of 16 angular states. The disk on which the main horizon scanning sensor's mirror is mounted has 16 slots which are sensed by an optocoupler which drives a counter. The counter is reset once per revolution of the disk by a separate sensor, so the angular states numbered 0 through 15 are sensed. When the photosensor detects the target light, the current state of the scanning angle is latched and can be read by the Z-80 mobile computer for transmission to the PDP-11 base computer.

its final approach moving in a forward direction.

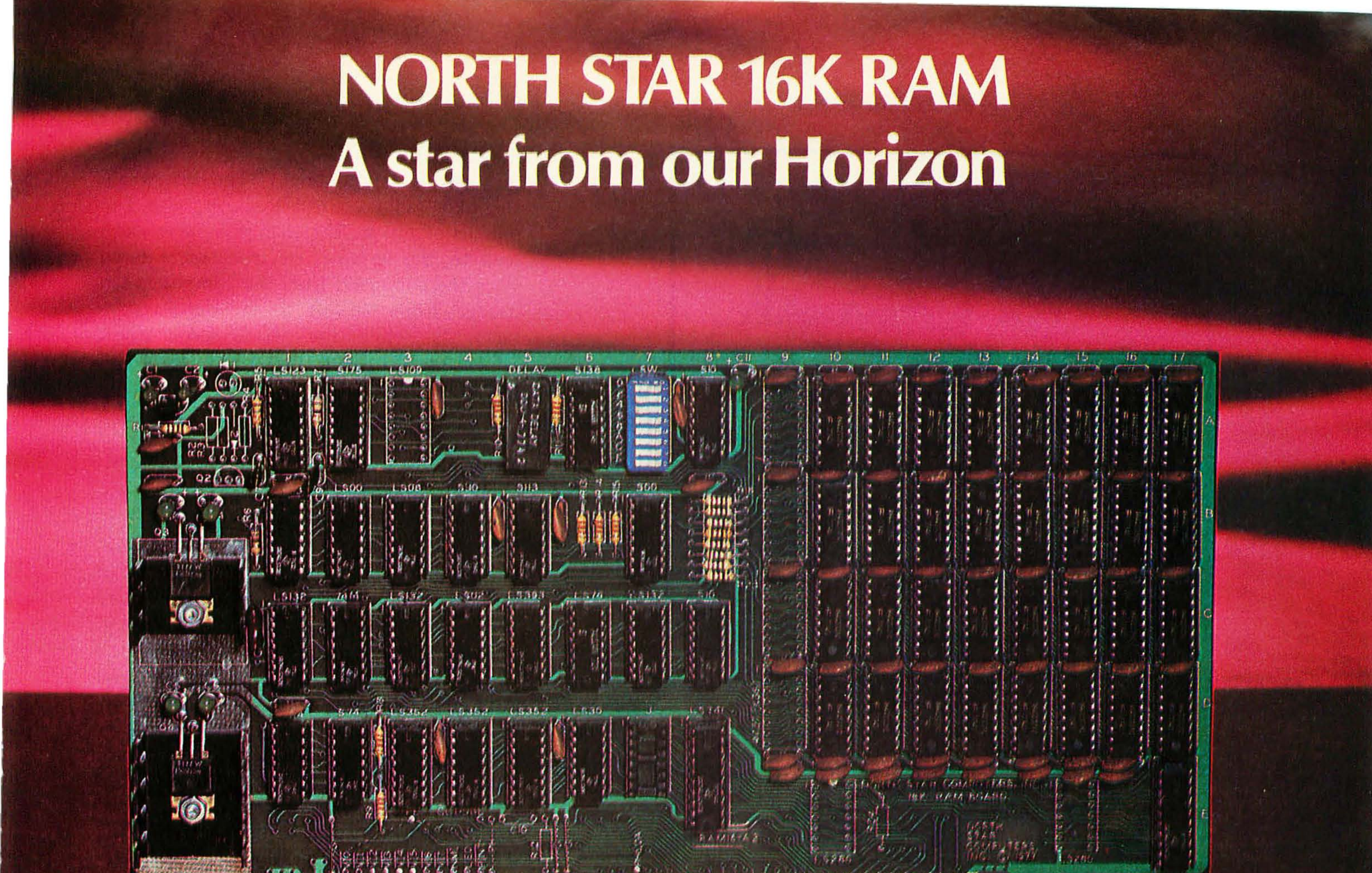
Sensors: Rotating Eye and Source Light Verifier

The rotating eye scans a plane about 18 inches above the ground looking for a light source. Photo 1 shows the physical arrangement. It automatically adjusts its sensitivity for ambient light, much like the human eye. The response of its electronics is such that it can detect a penlight at 30 feet in a dark room. In a normally lighted room it is self-adjusting and can discriminate between two lights if one is about three times as bright as the other.

As the disk rotates clockwise, the 16 slots in its edge pass through an optical switch and are counted by a 4 bit counter on the main deck. At the instant the light is spotted during the disk's rotation, the count is loaded into a 4 bit latch to be read by the computer. For example, if the light is spotted straight ahead, the count is eight. Figure 2 shows the logical definitions of the 16 possible directions (a missing slot corresponds to the state when the mirror is aimed to the rear; the counter is reset to zero in this condition). Thus any erroneous counts caused by ambiguous light sources or reflections are wiped out each time the disk begins a 360 degree scan. Once a number is loaded into the latch it stays there until the light is spotted again and a new number is loaded. This reloading usually occurs once for each time the disk goes around. But if the light source suddenly becomes blocked by some object, the latch continues to hold the last number loaded even though there is no light being seen. To remedy this problem, a source light verification circuit is part of the electronics. This circuit sends a logical 0 to the Z-80 as long as the light is actually still being spotted and a logical 1 when it is not.

Steering Control

The steering system has five possible positions numbered arbitrarily 2, 3, 4, 5 and 6. 2 is far left and 6 is far right. The command from the computer (PDP-11 or Z-80) has 3 bits to specify these states. The number is converted to an analog voltage using a current sourced resistor ladder. The DC voltage enables a pulse width modulator which controls the two steering servos. The servos act in opposite directions on opposite ends of the front axle to turn the wheels. The pulse is sent at 67 Hz. See figure 3 for a block diagram of the steering control section. If the pulse is 1 ms long, the servos stay where



The North Star 16K RAM board is a star performer in our HORIZON computer. Just as important, it is the ideal memory for most other S-100 bus systems. No other RAM board can surpass the speed, reliability, and quality features of the North Star 16K RAM at any price.

SPEED — The North Star 16K RAM is the fastest S-100 bus memory board available. No wait states are required, even with a Z80 at 4MHz. And, of course, this outstanding 16K RAM will operate with both 8080 and Z80 processors at 2MHz. Industry standard 200ns dynamic RAM chips are used. Invisible on-board refresh circuitry allows the processor to run at full speed.

RELIABILITY — The North Star 16K RAM is designed to match the same high standards as our MICRO DISK SYSTEM and HORIZON computer. For example, all address and data signals are fully buffered. A parity check option is available with the 16K RAM for applications requiring immediate hardware error detection. If a memory error occurs, a status flip/flop is set and an interrupt can inform the processor. Or, if preferred, an error status light will go on.

FEATURES — The North Star 16K RAM offers many desirable features. Addressability is switch-selectable to start at any 8K boundary. The board can perform bank switching for special software applications, such as time-sharing. Also, bank switching can be used to expand the amount of RAM beyond 64K bytes. Power consumption is minimal — the maximum power requirements are: .6A @ 8V; .4A @ +16V, and .1A @ −16V.

PRICES — $399 kit. $459 assembled, tested and burned-in. Parity option: $39 Kit. $59 assembled, tested and burned-in.

Write for free color catalog or visit your local computer store.

NORTH STAR ★ COMPUTERS
2547 Ninth Street • Berkeley, California 94710 • (415) 549-0858

Circle 285 on inquiry card.

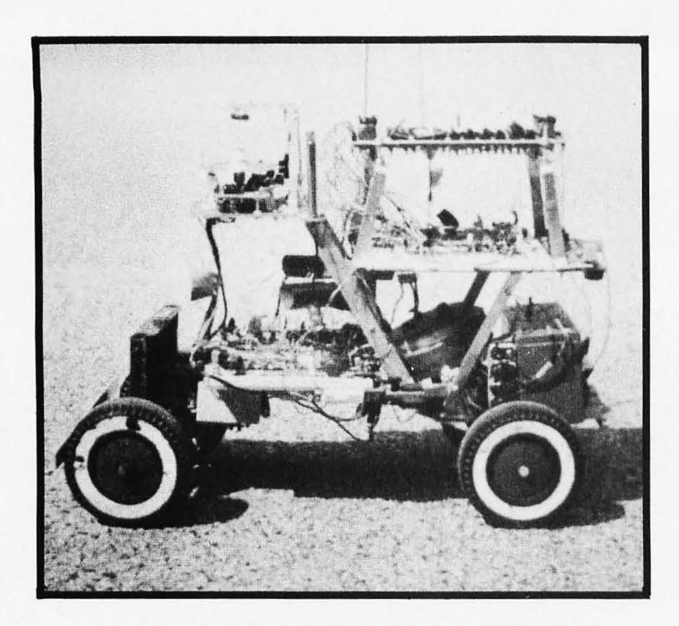


Photo 4: An overall view of the Tee Toddler taken from the side. Radio antennas and three levels of electronics on board are visible. The drive power source, a GeLi rechargeable battery, is at the right, with the front of the vehicle towards the left in this photograph.

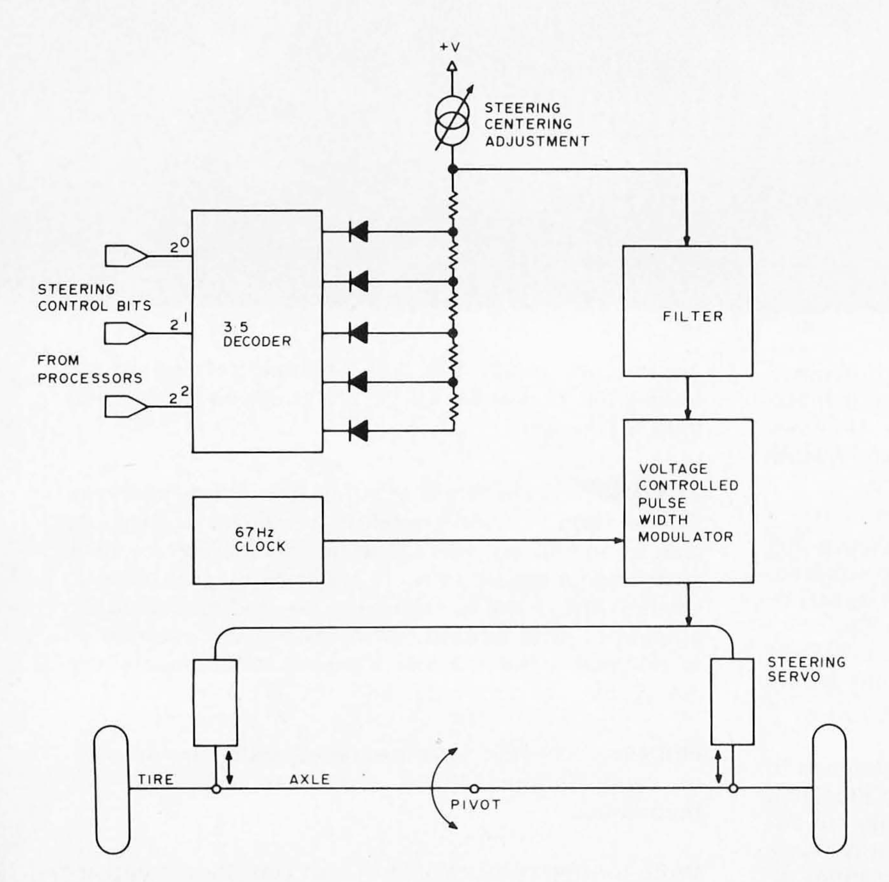


Figure 3: Steering system block diagram. Three bits from the Z-80 are decoded by a TTL decoder which implements a simple weighted resistor digital to analog converter. The output of the converter is filtered to prevent sudden changes and, in turn, sets the pulse width seen by the two servos. The servos are Heathkit radio control devices which have a 5 pound thrust and a 1.5 second full range response time.

they are; if it is longer or shorter, they move the wheels left or right. The servos are Heathkit radio control modules with five pounds of thrust. The change in pulse width seen by the servos is electronically filtered since the command from the computers can change from far left to far right instantaneously while the servos take about 1.5 seconds to pull the wheels from full left to full right.

Motor Control

The motor speeds are given by the numbers 0, 1, 2 and 3 which are decoded to stop, slow, medium and fast. There is also a forward and reverse bit, thus making a total of 7 motor states controlled by three binary digits. The motor control pulse width circuit works the same as the steering control circuit, except that the pulse change is not filtered. The motor is a 0 to 13 VDC "pancake" motor with a built-in 25:1 gear reduction. The rear axle differential gear ratio is 1:1. The motor's speed is controlled by the pulse width of the 12 V 700 Hz pulses being sent to it by the control circuit. Forward and reverse directions are controlled by a relay. The motor draws a current of about 1/2 A when the car is cruising at 1 mph (1.6 kmph). Since the motor is a highly reactive load to the sharp edges of the control pulses, the motor is optically isolated to eliminate interference with the logic circuits of the on board Z-80 system. Power amplification to drive the motor is accomplished after the isolator.

Sonar System

The sensing of objects in the car's path was originally intended to be done with light. This is difficult since objects with different textures at the same distance from the car would reflect different amounts of light. Pulsed infrared did not have the necessary intensity, and radar was ruled out because it would detect only metal objects. The existence of the National LM1812 sonar integrated circuit was probably the major factor enabling use of this sensor system. With this system the car is able to distinguish between obstacles to its left or right and can navigate between obstacles spaced only slightly farther apart than the car's width.

The sonar unit on the car transmits a 1 ms pulse at 40 kHz every 10 ms from a transducer mounted in the center of the car's front end (see photo 2). The echo is received separately on the right and left by two receiving transducers. Since there is

Collector's item

The first 16 issues of *BYTE* —

September, 1975 through December, 1976 —

in two gold embossed hard cover volumes.

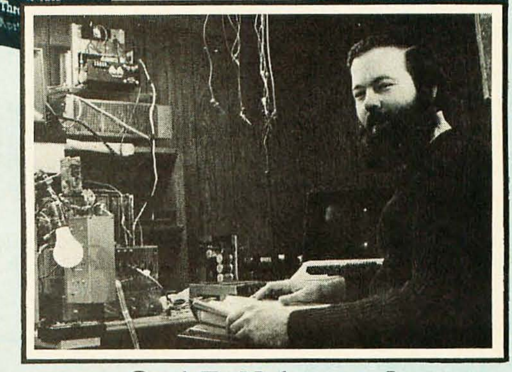
$100 $^{\underline{00}}$

complete set — two volumes

These individually numbered sets are personally signed by Carl T. Helmers, Jr., Editor-in-Chief of *BYTE*.

As only 100 sets will be offered for sale, orders will be handled strictly on a first-come first-served basis. Send in the coupon today! These early issues of *BYTE* are packed with articles by your favorite writers:
Helmers, Fylstra, Ryland, Dittrich, Chamberlin, Lancaster, Maurer, and many more.

For your personalized set of BYTE's first 16 issues mail this coupon today!



Carl T. Helmers, Jr.,
Editor-in-Chief, BYTE Magazine

Carl Helmers

Yes, rush me my numbered and personalized 2-volume set of the first 16 issues of BYTE.

☐ Check enclosed for $100.00

☐ Bill BankAmericard/Visa ☐ Bill Master Charge

Card Number ______ Expiration Date ______

Signature ______ Name *(please print)* ______

Address ______

City ______ State ______ Zip ______

Mail to: BYTE Publications, Inc.
70 Main Street * Peterborough * N.H. 03458

really only one sonar transceiver, the receiving transducers are multiplexed with field effect transistors to the receiver for three cycles of transmit and receive each. The count of the elapsed time between transmit and receive for each cycle is also multiplexed into the left or right output latch to be ready by one of the computers. A block diagram of the sonar system is shown in figure 4.

Since sound in air travels at about one foot per ms and a pulse is transmitted each ten ms, the maximum range is about five feet, which proves quite adequate. If no sonar echo is received, then the output latches are automatically loaded with the count 7, the maximum range possible. This is decoded by the computers as no object and thus no path correction is made. The sensitivity of the receiver is adjustable with a trimpot on the main deck, thus allowing different distance resolutions.

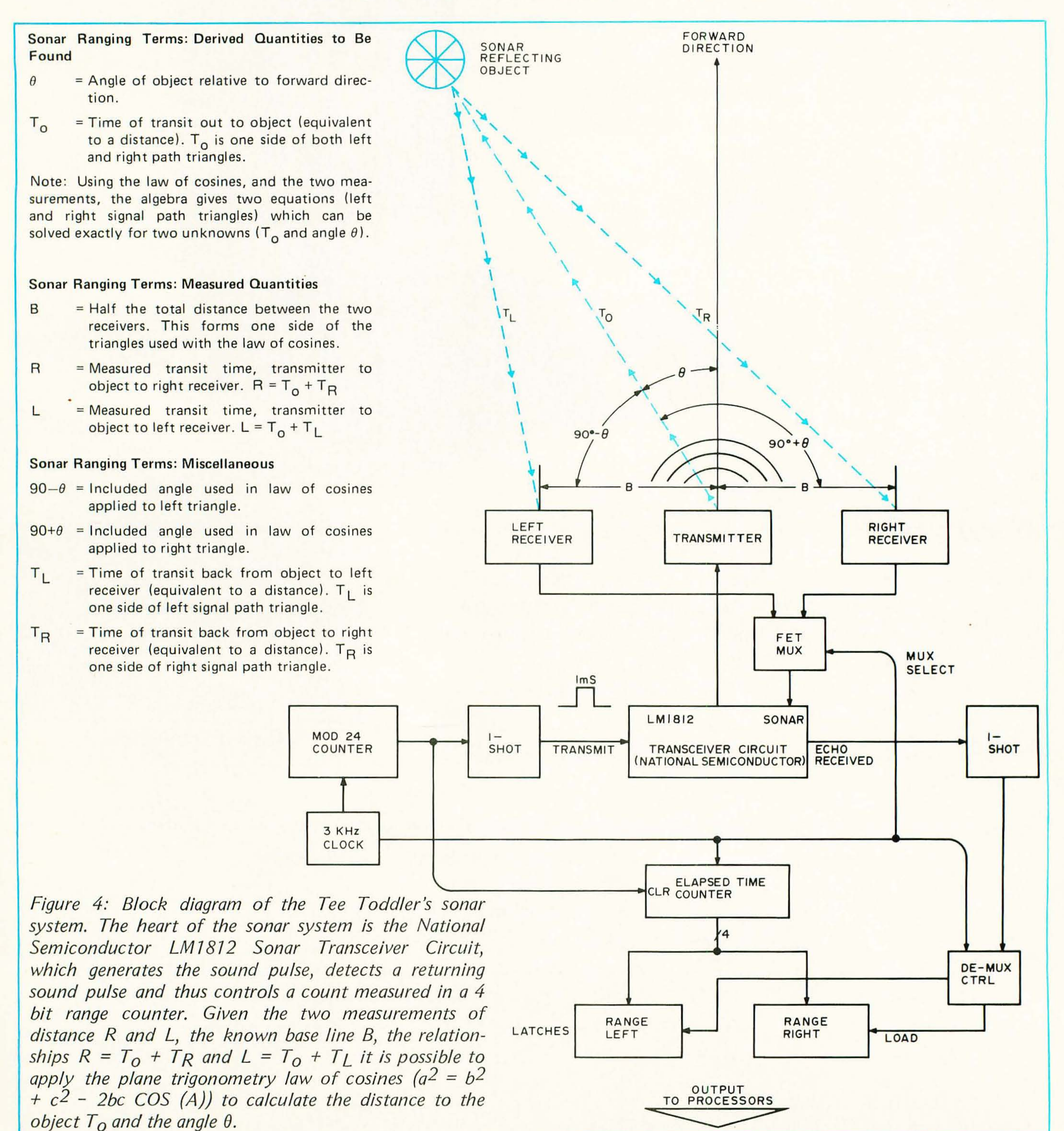


Figure 4: Block diagram of the Tee Toddler's sonar system. The heart of the sonar system is the National Semiconductor LM1812 Sonar Transceiver Circuit, which generates the sound pulse, detects a returning sound pulse and thus controls a count measured in a 4 bit range counter. Given the two measurements of distance R and L, the known base line B, the relationships $R = T_O + T_R$ and $L = T_O + T_L$ it is possible to apply the plane trigonometry law of cosines ($a^2 = b^2 + c^2 - 2bc\ COS\ (A)$) to calculate the distance to the object T_O and the angle θ.

INTRODUCING THE G/2 PROGRAM LIBRARY.

Now you don't have to be a programmer to use your computer!

At last there's a reliable, professional source of useful and entertaining programs for your personal computer. The G/2 Program Library from GRT unlocks the full power of your personal computer. Without any programming knowledge on your part.

The G/2 Program Library consists of applications programs pre-recorded on cassette tapes. A detailed manual with each cassette provides complete instructions. And G/2 programs contain prompting instructions. So they're easy to use every step of the way. Just load the tape and G/2 puts your computer to work.

The first three G/2 program tapes are available now: For exciting family games, get **Beat the House.** You'll find Blackjack, Craps, Roulette and Slot Machine programs, with truly random chances and realistic, casino-like payoffs.

Interested in your health? You'll want **Clinic.** Biorhythms, Dieting and Longevity programs give you new perspectives in health and happiness, along with some pleasant guidelines to follow.

Need a better handle on your money? Buy **Dollars & Sense.** Checkbook is a comprehensive program to balance your account and keep within your budget, while Best Choice is a novel way to make decisions based on the facts you know and the relative importance of each. Use it for everything from choosing a new car to selecting your next stereo.

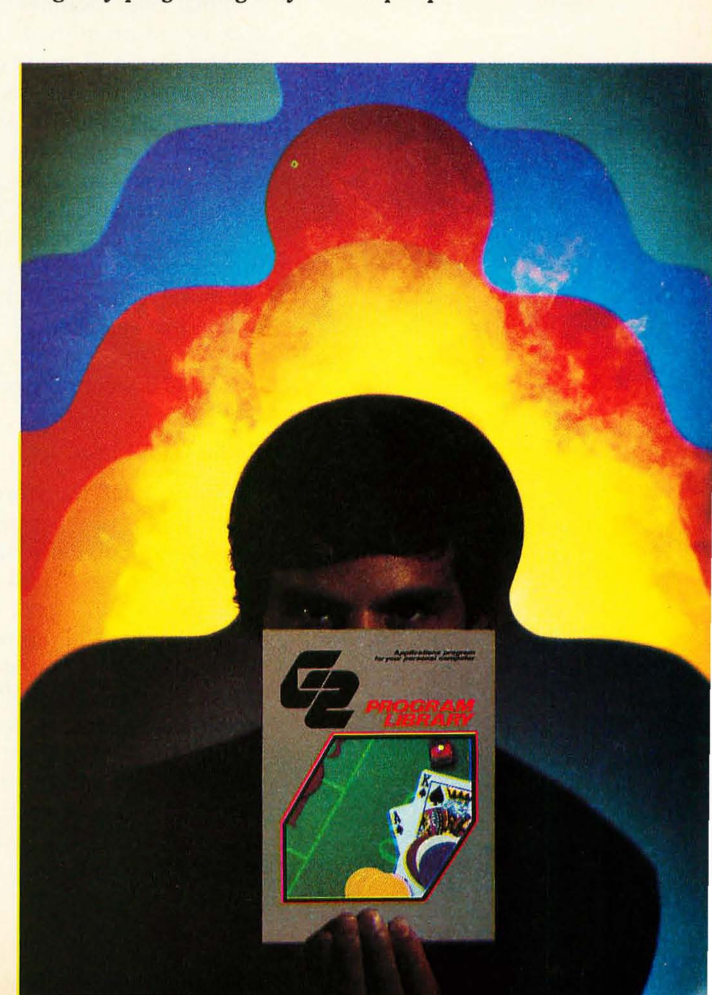
Soon, every dealer that handles personal computers will stock the G/2 Program Library. For the G/2 dealer nearest you, call us toll-free at 800/538-1770. (In California, 800/662-9810.) And if your favorite computer dealer doesn't yet offer G/2, have him call the same number.

Soon you'll see G/2 program tapes in many other fields of interest. The G/2 Program Library is now available for use with SOL, SWTPC, Exidy, TRS-80 and Apple personal computers. Just select the package marked for the computer you own.

Start building your G/2 Program Library now. And put the full power of your computer to work for you.

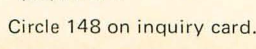
THE REASON YOU BOUGHT YOUR COMPUTER.







A product of GRT Corporation
Consumer Computer Group
1286 North Lawrence Station Road
Sunnyvale, California 94086
(408) 734-2910



Circle 148 on inquiry card.

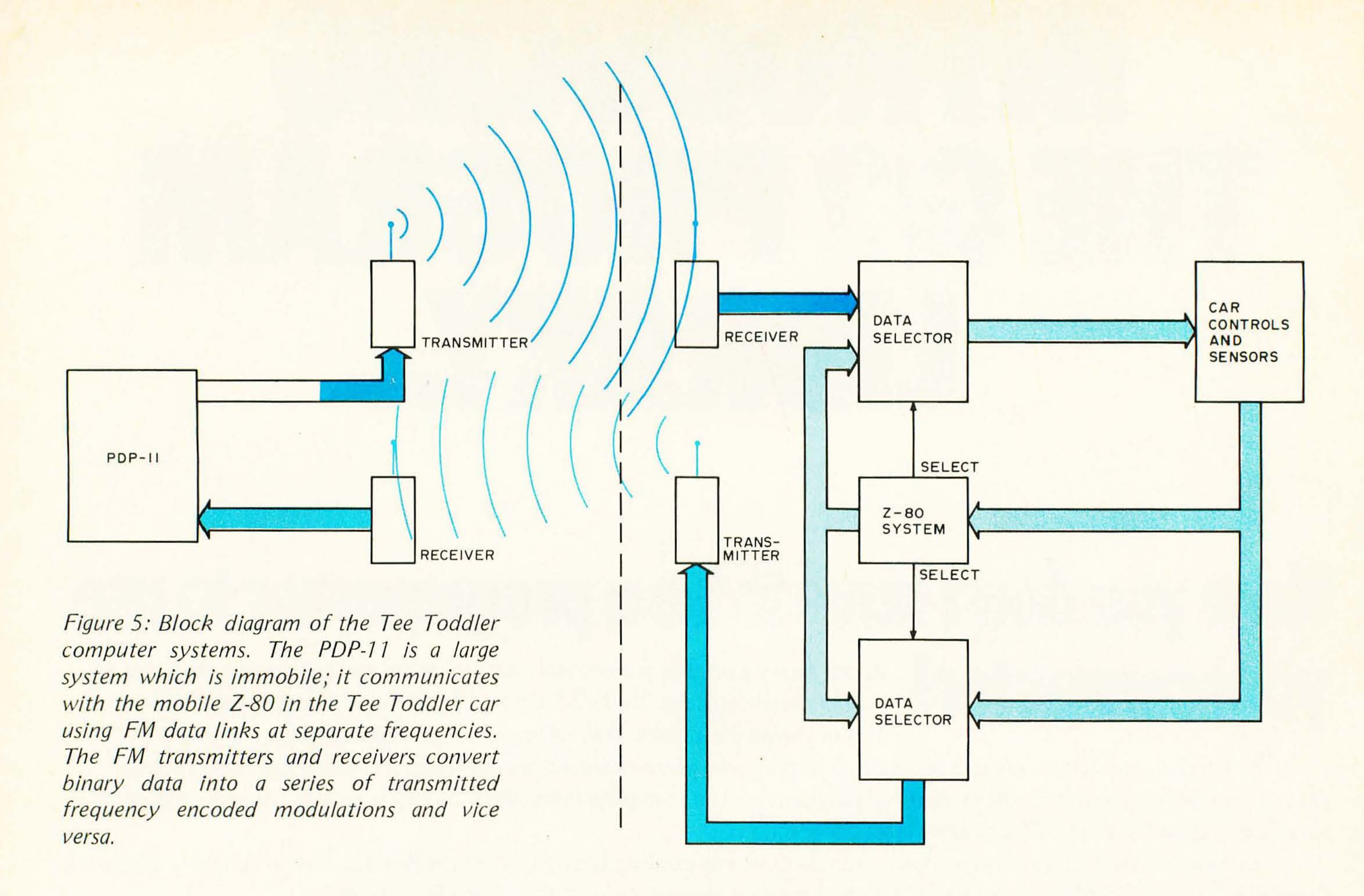


Figure 5: Block diagram of the Tee Toddler computer systems. The PDP-11 is a large system which is immobile; it communicates with the mobile Z-80 in the Tee Toddler car using FM data links at separate frequencies. The FM transmitters and receivers convert binary data into a series of transmitted frequency encoded modulations and vice versa.

This type of sonar system is really a minimal one. Doppler shift detection could also be accomplished fairly easily to allow determination of speed of a moving obstacle. Echo amplitude analysis would also be worth investigating since it would help solve the problem of echo frame overlap; such overlap exists when an echo from the previous sounding returns late, after bouncing off a far away object, resulting in two echos for the current frame. The strongest of the two (or more) echos should be taken as the true one. The Tee Toddler's system triggers a 9 ms oneshot on the first echo thereby ignoring all secondary echos.

Radio Data Links

An encoder transforms parallel bits into serial tones to be transmitted over a frequency modulated channel. One channel is from the Tee Toddler car to the PDP-11 computer at 96 MHz. The other channel is from the PDP-11 computer to the car at 450 MHz. The serial data encoder and transmitter at the PDP-11 base station are essentially identical to the car's versions except for the number of bits per word of data. A two-tone modulation system is used. This means that each binary state is encoded into one of two different frequencies for transmission. At the modulator a logical 1 is represented by a 2500 Hz signal and a logical 0 is represented by a 1900 Hz signal.

The receiver and data decoder accepts the string of audio tones from the FM receiver, decodes them into 1s and 0s using phase locked loops and converts back to a parallel data format. While our prototype did not use standard circuitry, a standard asynchronous serial communications discipline such as that provided by a UART or ACIA would work well in this application.

Power Sources

The power for most circuits is derived from a 12 V 4.5 Amp-Hour GeLi cell rechargeable battery. The battery was drilled and tapped at 8 V to power a 5 V regulator for the TTL circuits and for the Z-80 microcomputer. The steering servos required their own set of four penlight batteries (also rechargeable), and the 1702 read only memory holding the Z-80 program required a separated −9 V supply, derived from three parallel transistor radio batteries. This power supply system is capable of running the car for several hours before any recharging is necessary.

Computer Control

As has been mentioned, the car is con-

trolled by two separate computers. The communication paths between the computers and the car are shown in the system diagram of figure 5. In actuality, only one of the computers can communicate with the car at a time. This is the case for several reasons. First, there are only three control inputs to the car to make it operate. These are the speed, direction and steering controls. Since these inputs can originate at either computer, a multiplexing scheme had to be used. Second, only the PDP-11 actually makes decisions based on sensor information from the car. The Z-80's control of the car's movements is more like a reflex action, in that it performs a canned routine when invoked by the car's sensors. Last, the functions are separated to facilitate the transition to a total on board control system, since the PDP-11 can be replaced easily by another on board microcomputer.

The motivation behind this configuration is based on several criteria. Since part of the system was going to be standing alone, some of the major considerations were power consumption, various power supply requirements and ease of operation. With all these considered, it was decided that a Z-80 with its single 5 V power supply requirement and single phase clock was a logical candidate. The 16 bit PDP-11 was used because it could do computations at a greater speed than the 8 bit Z-80.

The on board microprocessor has several functions associated with the control of the car. One function is to supervise all data and control channels to and from the car. In other words, it has the responsibility of deciding whether the PDP-11 or the Z-80 is going to control the movements of the car and which of the two computers is going to receive the information from the car's sensors. The routing of these different channels of information is accomplished by the use of data selectors. The Z-80 controls the data selectors such that the information is routed to the correct device at the correct time. Information coming in to the car to control its movements comes from either the PDP-11 or the microprocessor. It comes from the PDP-11, over the radio link, if the car's sensors indicate one of the following conditions:

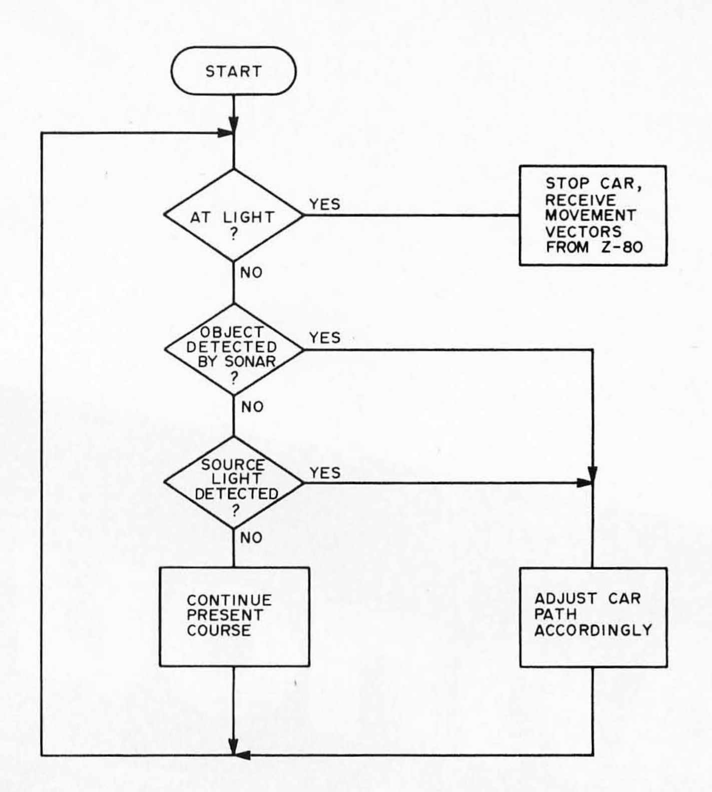
- The car has reached the source light.
- An object has been detected by the sonar system on either the left or the right.
- The car has spotted the source light.

The PDP-11 then analyzes these conditions according to the hierarchy of importance, as is shown in the decision tree of algorithm 1, and then communicates to the car the appropriate movement corrections to make. Control information to the car originates from the Z-80 when one of the following car sensor conditions arises:

- Contact with an object has been indicated on either the left or right side by the front bumper.
- The car has lost sight of the source light.

The microcomputer controls the movements of the car if either condition is met and then gives control back to the PDP-11 when it has finished its corresponding task.

Another function of the on board microcomputer is to store all movement vectors associated with the car's path. These vectors indicate the steering angle, the direction of travel and the length of travel of the car. Therefore, when the car changes direction or steering angle, a vector is stored in memory which correlates to how far the car traveled at the previous setting. Thus, when the task of finding the light is accomplished, the on board memory contains all the different moves the car made to reach



Algorithm 1: The base computer's executive program in outline form. This decision tree is executed in the PDP-11 each time a car sensor word is received. If any of the tests results in an affirmative answer, the program executes a routine designed for that specific state. Each routine takes into account past information of where the light was spotted. The sonar detection routines also take into account any objects which have recently been passed. These things are considered so that the car proceeds in the direction of the light and does not collide with any objects while moving in reverse. There is no specific way to stop the system except by interrupting the PDP-11 and issuing a control word to the car to stop it.

Announcing

Hazeltine proudly announces our newest and lowest-price addition to the Leadership Line of Economy Terminals, a worthy companion to the famous Hazeltine 1500 Series.

It takes a major High-Technology Company like Hazeltine to:

- Effectively apply advanced microprocessor technology to achieve the 1400's Ultra-High Reliability.
- Implement unrelenting QA/production procedures to achieve the 1400's Ultra-High Quality.
- Design and build a terminal that delivers More Performance Per Dollar than any other TTY-compatible terminal now on the market.
- Meet the growing needs of the video terminal markets—OEM, End-User and Personal Computing.

Hazeltine
Hazeltine 1400 Series
performance in economy terminals

Hazeltine 1400

Now

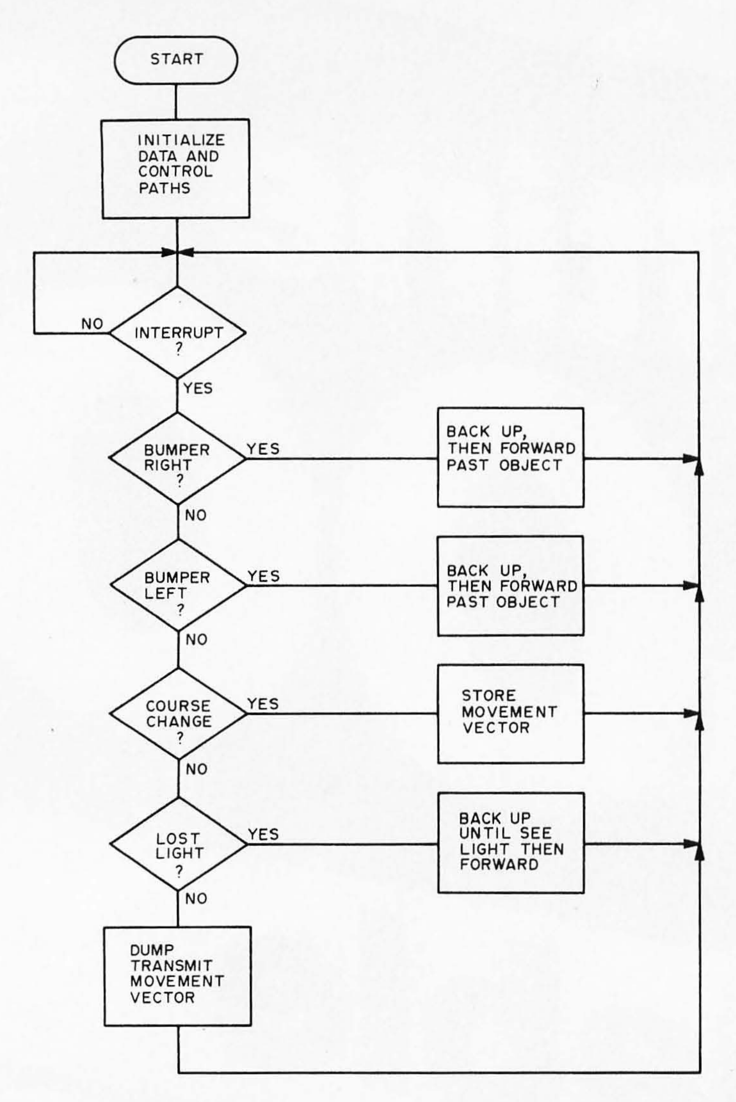
Hazeltine 1400 Available



Hazeltine Corporation, Computer Terminal Equipment, Greenlawn, N.Y. 11740 (516) 549-8800 Telex 96-1435

EAST: N.Y. (212) 586-1970 □ CENTRAL: Chicago (312) 986-1414 □ SOUTH: Atlanta (404) 952-8444 □ WEST: Burlingame (S.F.) (415) 342-6070
ENGLAND: Hazeltine Ltd. 01-948-3111 Telex (851)-928572 □ GERMANY: Hazeltine GmbH 0611-590748 Telex (841)-416924
FOR WORLDWIDE SALES INFORMATION CALL: (516) 549-8800

Circle 156 on inquiry card.



Algorithm 2: The mobile computer's executive program, in outline form. In the initialization procedure the Z-80 sets up the memory areas, resets the wheel rotation counter and sets the data selectors for the PDP-11 to control the car and the car sensor status word to be transmitted to the PDP-11. The processor then awaits an interrupt. When the Z-80 is interrupted, the tests are executed in this order. After each routine is completed, control is returned to the PDP-11.

the light. With this information the car has a new path calculated for it by the minicomputer. This path allows the car to drive to the light from the same starting position without the use of any of its sensors and without having to maneuver around a single object, since it already knows where they are located. In a very loose sense then, the system has learned about its environment and used this knowledge to improve its proficiency at the task of finding the source light, much like a mouse in a maze.

The last major function of the microcomputer is to pass the movement vectors to the PDP-11 once the car has reached its destination. This is accomplished by having the minicomputer issue a "dump" command signal over the radio to the Z-80. Using a handshake system, the Z-80 sends the movement vectors to the PDP-11 one at a time. The decision hierarchy of the Z-80 program is shown in algorithm 2.

PDP-11 Base

The PDP-11 minicomputer is the actual brain of the system. It has the ability to decide where to move the car in order to approach the light and yet avoid objects on the way. Inputs to the PDP-11 come from either the car's sensors or the microcomputer memory. If the inputs are from the car they indicate the current status of the sonar left and sonar right sensors, the 360 degree rotating eye and the source light intensity monitor. These are processed according to the following hierarchy. First the source light intensity monitor signal is checked to determine if the car has reached its destination. This indicator is checked first because it will indicate if the total task has been accomplished. If this condition is true, the PDP-11 computer sends a message to the car telling it to stop and telling the Z-80 to start unloading its memory of movement vectors. The handshake system used is initiated by the car informing the PDP-11 base computer that the car has reached the light. The minicomputer then informs the microcomputer to start unloading the memory, at which time the minicomputer checks each incoming vector to determine if it is the last. If not, the PDP-11 asks for another vector to be transferred. This continues until all vectors are transferred.

Second, the sonar sensor inputs are examined to see if any objects are being detected. If an object is detected, then a special routine analyzes the situation according to which side the object is detected and how far away it is. If the object is far enough away for the car to maneuver around it without having to back up, then the PDP-11 commands the car to steer to the left or right, whichever is appropriate, to avoid the object which is in the way. If the object is detected by both sensors, then the side which detects it as being closer overrides the other. In the event that the distance measurements are equal, the computer arbitrarily chooses the right side as having a higher priority. Obstacles detected at a range too close for the car to maneuver around while proceeding forward cause the car to back up. An obstacle detected at a very close range on the right causes the car to back up. However, the steering position for this movement depends on whether the car was steering to the left, right or center. If the car was proceeding to the right, then it must know, from a previous sensor reading, that the source light is to the right. If this is the case, the car backs up

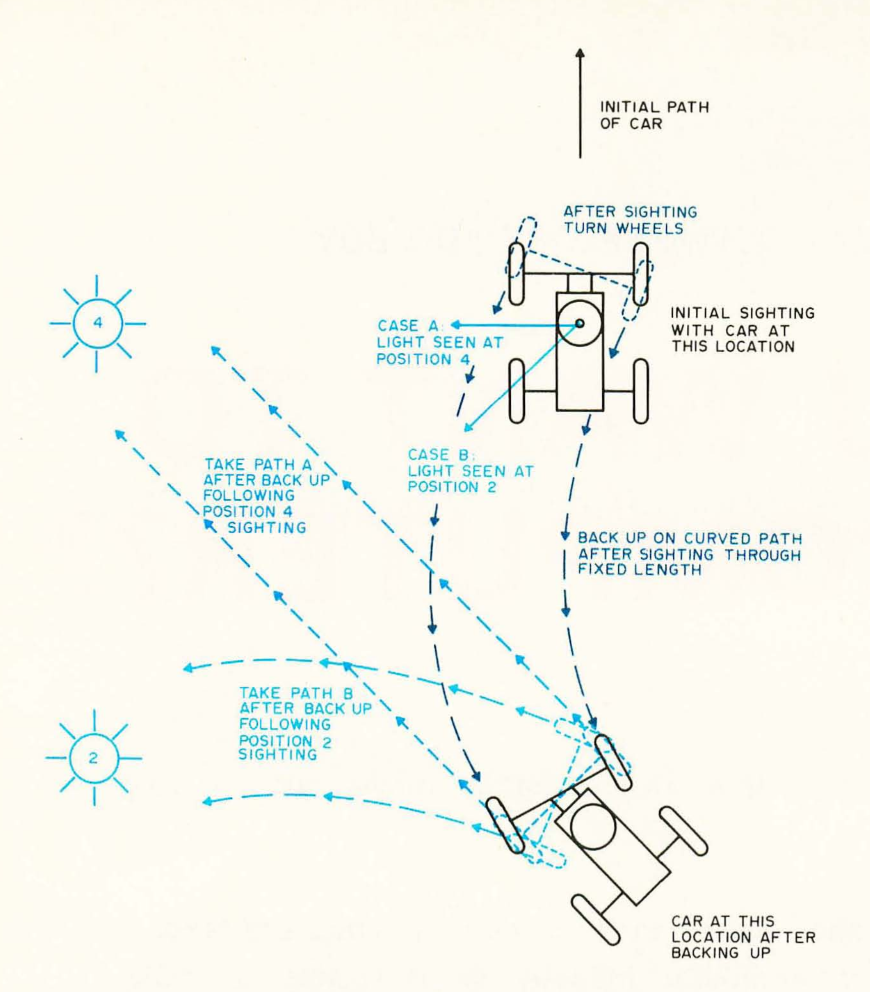


Figure 6: This diagram illustrates paths the car can make when confronted with typical situations. Assume the car is moving straight and forward and detects the light in either the number 2 or 4 position. The PDP-11 has control and moves the steering to the right and backs up for a certain distance. If the light was spotted in position 4 the car proceeds along path A with the steering set back to the center. But if the light was spotted at position 2 the car moves along path B with the steering set to the left. If the light had been spotted at position 1, almost directly behind the car, the maneuver would also have been along path B. However, since it would not be able to approach the light at a very straight angle, it would have to back up again and set the steering for a more direct path.

with the steering set to the left. After backing up for a certain amount of time, the car changes the steering to the right and proceeds forward. This causes the car to maneuver around the obstacle and also sets it on a better path to the source light. In effect, the car has used past knowledge to evaluate the present situation. If the steering had been to the left originally, then the computer would set the steering to the right. The car then proceeds in reverse for a given amount of time and then changes its steering position to the left and proceeds forward for an additional period of time. This action makes the car maneuver around the object and along a better path. An original steering position in the center again causes the right side to override the left. Although all these controls and decisions are handled by the minicomputer, the results do incorporate the use of the on board microcomputer. The microcomputer is used to store all movement vectors pertaining to all direction or steering changes. This is accomplished by having the PDP-11 computer issue a course change signal to the Z-80 at the same time it issues the new control word to the car. The Z-80 on board computer then stores the previous movement vector in memory, then returns control to the minicomputer.

The third sensor readings used by the PDP-11 are the values from the rotating eye. These sensors indicate where the light is located with respect to the current position of the car as shown in figure 2. Basically, the world, as the car sees it, has been divided into 16 windows, each allowing a different view of the horizon. A number has been assigned to each separate slice, thus giving an easy identification and recognition scheme for evaluating the position of the source light. For example, the semicircle in front of the car has nine windows associated with it, four to the left, four to the right, and one for the center. Thus, since straight ahead has been declared as having a value of 8, then the far left becomes 4 and the far right becomes 12. Therefore, by examining the value, the computer can tell where the source light is and then adjust the path to proceed in that direction. If the light is spotted in the forward semicircle between values 5 and 11, the course adjustments are quite straightforward. The steering is merely positioned so as to point the car in the direction of the light.

However, if the light is spotted in the aft semicircle or to the extreme sides, a different approach must be taken. Instead of having the car do a complete 180 degree turn, we decided to have the car perform several backward and forward movements. By doing this we reduced the possibility of contacting objects by reducing the space needed to perform the maneuvers. For example, if the light is detected in the number 1 through 4 windows, the computer backs the car up with the steering set to the right. The distance it backs up depends on which window the light was spotted in. For example, if it was spotted at position 4, the extreme left, then the car would back up far enough so that when it stopped it would be facing directly toward the light and could then proceed in a straightforward direction (path A in figure 6). If, however, the light had been spotted more to the rear, the car would back up a bit further and then have the steering set to the left position and proceed forward. The exact opposite action would have taken place had the light been

THE BOOK YOU'VE WAITED FOR IS HERE!

What is an accumulator?

What's an I/O port?

What are EPROMS?

Does a program counter count programs?

How do you use relative addressing? Indexed addressing? Boolean logic operations?

What is paging? A stack?

What's the difference between a dot and vector CRT display?

How do you copy programs using an ordinary tape recorder?

What can a microcomputer really do?

What kind of personal computer should I buy?

UNDERSTANDING MICROCOMPUTERS answers these and hundreds of other questions clearly in this profusely illustrated, easy-reading "must" book. You'll find fundamental concepts behind the operation of virtually all microcomputers, written in simple English so that anyone can gain the extra knowledge needed to read **and understand** computer magazines, manufacturers' literature, and feel "at home" around computers. Here are just a few items discussed...

- How a CPU is organized and how it follows sequences of directives to solve problems
- Illustrates basic instructions from almost every class of microprocessor
- Discusses common memory addressing modes—illustrates typical uses
- What you must know to tell a computer what to do when using machine language programming methods
- Explains use of flow charts, program worksheets, hand assembly of source codes into object codes, memory maps, purpose of an Editor, Assembler, Monitor
- Benefits of using a high level language
- Introduction to BASIC
- How a computer communicates. Types of commonly used input/output devices and their operational concepts
- Practical aspects of selecting a small computer system
- Plus, hundreds of other practical facts and information

If you are at all curious about small computers, you must own this 300 page no-nonsense text that reads like a simple book! Includes an easy-to-use glossary of key microcomputer oriented words. **UNDERSTANDING MICROCOMPUTERS.** The name says it all. Order your copy now!



SCELBI COMPUTER CONSULTING INC.
Dept. 000
Post Office Box 133 PP STN
Milford, CT 06460

Price shown for North American customers. Master Charge, Postal and Bank Money Orders preferred. Personal checks delay shipping up to 4 weeks. Pricing, specifications, availability subject to change without notice. SCELBI Books are available in many fine Computer Stores.

IMPORTANT! Include 75¢ postage/handling for each item delivered by U.S. Mail Book Rate; or $2 for each item shipped First Class or via UPS.

Now available in hardcover for only $14.95... and paperback for only $9.95. Order yours today!

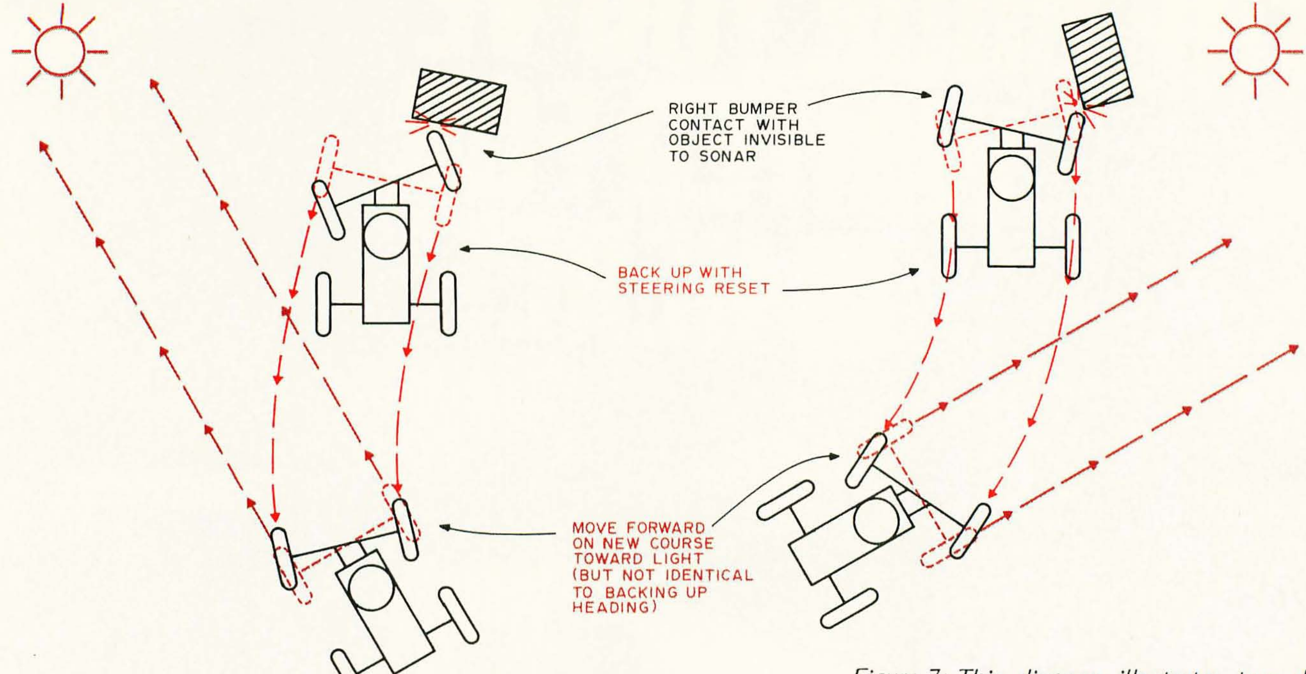


Figure 7: This diagram illustrates two different situations of the car having contact on the right with an obstacle. The obstacle size is exaggerated for the sake of illustration. The diagram on the left depicts the car steering to the left toward the light when it strikes the object. The Z-80 takes control and adjusts the steering to the right and backs the car up a certain distance. The steering is then set to the center position and the car proceeds forward. On the right, the car is proceeding to the right toward the light. Here the Z-80 sets the steering to the left, backs the car up then adjusts the steering to the center and proceeds forward. In each instance the car maneuvers around the obstacle and on a path toward the light.

spotted on the opposite side. Once again the on board computer would have been interrupted to store all course changes associated with the maneuvers. The problem of running into objects while proceeding in reverse is actually of minimal concern due to this method of reaching the light when it is spotted in the rear. If there had been an object there it presumably would have been detected and the path adjusted accordingly. However, this adjustment would not have been made without taking into account where the light was being spotted. Since this is the case, it is not possible for the car to be turning left or right in a forward direction if the light is actually behind the car. If, however, the light has not been spotted yet, then it is feasible that the car can run into something in its reverse move, since no attempt is being made to look back into the movement vector memory to determine if an object has just been maneuvered around.

The final function of the minicomputer is to stop the car once it has reached the source light. This is accomplished by detecting the source light intensity monitor bit as it changes to the active state. Once this occurs the minicomputer stops the car and at the same time initiates the handshake operation with the on board computer to start the transfer of the movement vectors. The minicomputer stores these vectors as they come over the radio until all have been passed. The last vector is actually a null vector, or all zero, which indicates all vectors are transferred. The minicomputer now does one of two things. It can automatically plot out a new course for the car to take, or it can display the vectors graphically on a screen. With the latter method the user is able to see all the moves and recalculate a new path himself based on his visual perception of the path taken. The automatic method is simply a sequential analysis of the vectors by the computer. If the car makes a move in reverse the computer assumes that either an object was detected or the light was spotted behind the car. In either case the computer adjusts a move previous to this occurrence, thus allowing the car to anticipate the upcoming situation and act in accordance with the situation. By adjusting these movements prior to detecting the need to reverse direction the computer has eliminated this need altogether and has thus "curved out" the path, making broad sweeping turns as opposed to jerky forward and backward movements.

Microcomputer Functions

The on board Z-80 computer provides the reflexes and signal control for the whole system. In the event that the car hits a thin object which is not detected by the sonar, a reflex action is invoked, much like a human response to a given stimulus. The various reflex actions this computer controls are: loss of sight of the light and touch stimulus from either sides of the bumper. To initiate a microcomputer routine for either the reflex actions or the control functions, one of the following interrupt inputs must become active:

- A signal from the front bumper.
- A signal indicating loss of the source light.
- A course change.
- A request to dump the movement vectors.

All these signals are ORd together, thus enabling any one of them to initiate an interrupt. When the Z-80 is interrupted it interrogates an external buffer to determine which condition caused the interrupt. The program (see algorithm 2) then checks each bit, one at a time, to determine which one is active. If more than one is active, it only processes the first one checked which is active. If none of the lines is active, then the program defaults to the dump line being active. Upon determining which stimulus is active the program executes a specific routine for that particular interrupt.

The bumper right and bumper left routines are essentially the same except for the steering positions being reversed. If contact is detected with the right side of the bumper, the Z-80 receives an interrupt and the car automatically backs up. Figure 7 illustrates the bumper reflex. The direction in which it backs up depends on the direction it was travelling when it collided with the object. If the steering was set to the right, then it must have previously detected the source light to the right (see right illustration in figure 7). Then, in order to maintain this general direction, the car backs up with the steering set to the left. After backing up for a certain time the car sets the steering to the right and proceeds forward past the object and toward the light. Although this setting is not a direct heading toward the light it is in the general direction and it has avoided the object.

If the direction of travel was to the left, (left illustration, figure 7) then the car backs up with the steering to the right and then proceeds forward with the steering set to the center. All steering actions would simply be reversed for a contact on the left. Therefore it is easy to see that all the reactions to the stimuli are preprogrammed and always net the same result, thus they exhibit a reflex action.

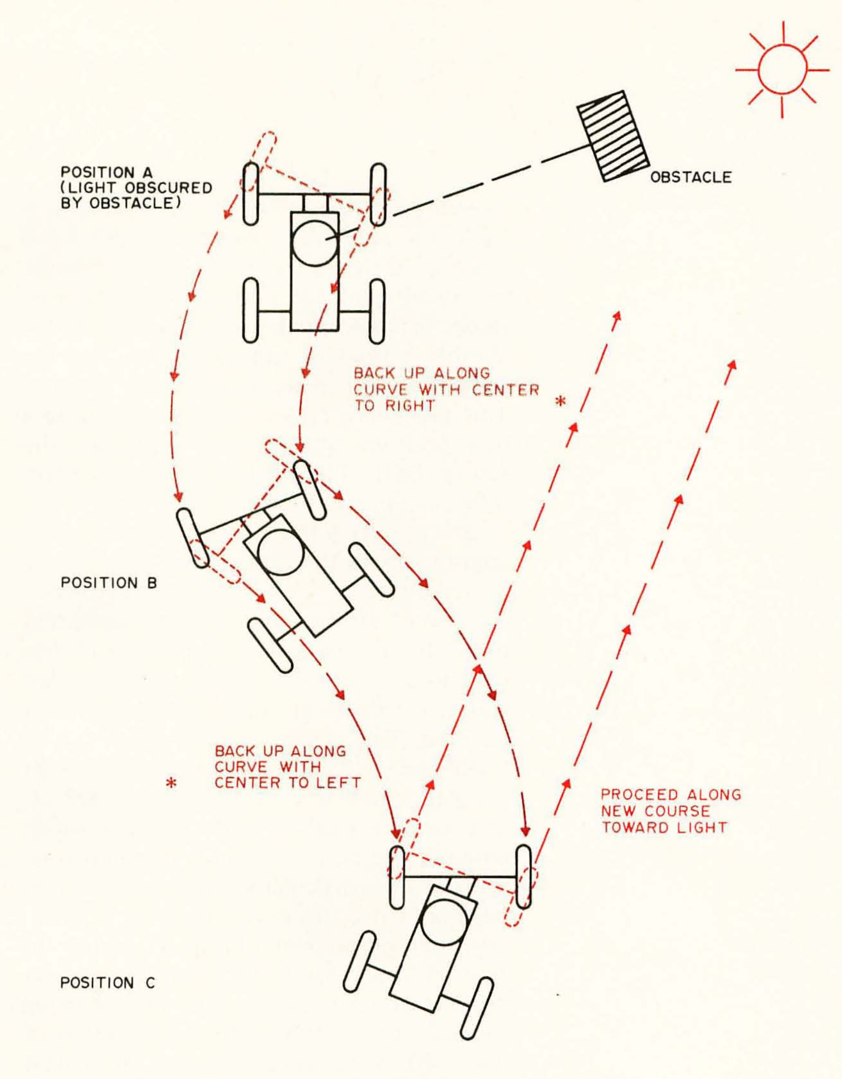


Figure 8: This diagram illustrates the Z-80 reflex for the case in which the car has lost sight of the light at position A. The Z-80 instructs the car to do a reverse S turn. First it adjusts the steering to begin the turn and travel to position B. Then it adjusts the steering to have a center of curvature to the left and continues reverse travel to position C. At the end of the path, the steering is again set to the appropriate position for a course towards the light and the car resumes forward travel without its goal being obscured by the obstacle.

The other reflex action is quite similar (see figure 8). If the car is travelling in any direction and loses sight of the source light, then apparently what has happened is that an object has come between the car and the light, thus obscuring the car's "vision" as at position A in figure 8. Although the car knows the object is there, it cannot detect

its exact location. What the car does then is back up with the steering in the same position, then after a certain time it changes the steering to the opposite side and continues to back up. It continues backing for another period of time and then sets the steering to the center and proceeds forward. This procedure causes the car to move to a new position where it can again see the source light. This again exhibits a certain reflex action controlled by the Z-80.

Since these reflex routines involve many adjustments to the car's path, it is necessary to record all of these separate movements. Therefore, at the end of each segmented move the routines call on the course change routine to record the current wheel rotation count, direction of travel and the steering position. This course change routine can be called on from either the Z-80 through another routine or directly from the PDP-11. The routine reads a buffer register which contains the current number of wheel rotations at this particular steering and direction position. Once this vector is stored in memory, the routine resets the wheel counter to zero in order for it to count the correct number of revolutions at the next steering and direction setting. The last function of the Z-80 is to transfer all the movement vectors from the on board memory to the PDP-11. As has been discussed, the Z-80 responds to a request from the PDP-11 by sending the vectors on a last in first out basis, one at a time in correspondence with the handshake system. Once all vectors are passed the Z-80 reinitializes the car and passes control to the minicomputer.

After any of these routines has been processed the Z-80 returns control of the car to the PDP-11.

Computer Design Specifics

The microcomputer designed for this robotic application is equipped with only the bare essentials. The total system consists of 256 bytes of programmable read only memory, 1 K bytes of volatile programmable memory, three bidirectional IO ports, one Z-80 microprocessor and one 8 bit line driver. The read only memory is a 1702 UV eraseable part which contains the program. The programmable memory is made up of eight 2102 parts. The IO chip is an Intel 8255 and was chosen because of the number of ports available.

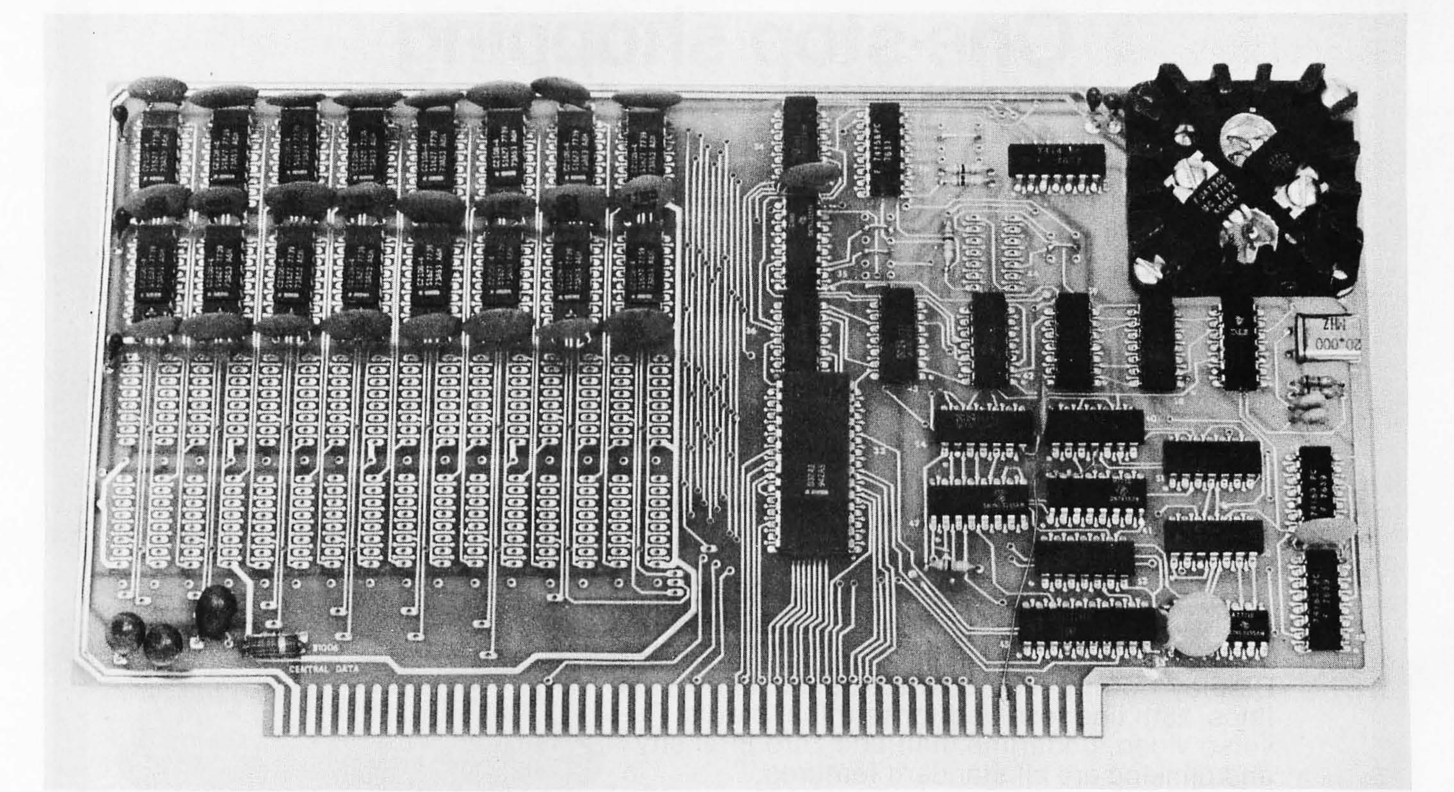
When designing a dedicated system like this, one must keep in mind that the probability of it working the first time is very nearly zero. Therefore, care must be taken to make the system as easy to debug as possible. This system was designed with this in mind and thus several additional functions were included in the design. A reset switch is installed on the computer board to aid in checking different functions of the system under the same circumstances. A single step switch is also located on the board. By using this, one can step through the program and examine different signals to determine their validity. The line driver was installed specifically to allow the examination of the data lines. Included in the design are provisions for the addition of 1702 memory chips up to a total of 1 K bytes of program memory. Since the system was of a prototype nature it was built on a perforated board and was wire wrapped. Care was also taken in the use of the PDP-11. Before the final application program was written several simple test programs were written which checked out the two way radio links and the responses of the car to commands. With the test programs it was possible to enter commands at a terminal and control the actions of the car in a remote control fashion as well as to receive a continuous read out of the current status of the car's sensors which are used by the minicomputer. This proved to be one of the invaluable debugging aids of the overall system.

Concluding Comments

Projects involved with robotics are a logical extension of microcomputer technology. The possibilities of building such dedicated "artificially intelligent" machines are almost limitless. It is not unreasonable to think that personal computer experimenters could build a robotic machine at home. However, a little forethought is worth a lot of time and effort in the end. Think about what the machine is going to do, and what is necessary to accomplish this. Build the system in modules which are easy to interface to each other and also easy to debug and repair. The capabilities of the machine are only bounded by the imagination of the designer. Perhaps the ultimate goal of a robotic machine is to have it perform its designated task consistently.

We think that Tee Toddler has proved an adequate fulfillment of that goal in the limited context of a light seeking mobile device. ■

Memory Makes the Computer, Central Data Makes the Memory



Memory is an important part of a computer system. So don't even think of settling for a poor quality RAM board. Central Data offers carefully engineered 16K, 24K and 32K RAM boards to suit your needs. We take our RAM boards seriously, so should you.

Reliability. Each one of our boards comes completely assembled, tested and burned in. No fiddling with an infinite number of parts from a kit or costly investment in equipment to test your board. Each one of our boards is backed by a one year warranty to insure your satisfaction.

Expandability. We offer a 16K RAM board with capacity to 32K. You can purchase a 16K board and expand later or buy a full 32K.

Performance. Unlike some dynamic memory boards, ours uses an invisible refresh that inserts refresh cycles between your computer's normal memory access cycles. This eliminates loss in performance of your computer with lengthy refresh cycles.

Power. Our board uses less than one-half the power of a static RAM.

Price. You can buy a 16K RAM board, completely ready to use, for only $289. A full 32K RAM board is available for $475, or you can add on later for only $200.

When you buy our memory board, you'll be buying the best equipment at the best price. Order yours today by mail or phone.

Please send me the following:
☐ 16K RAM board ($289)
☐ 32K RAM board ($475)
☐ additional information
I enclose $__________.
Ill. residents add sales tax.
Central Data P.O. Box 2484, Sta. A Champaign, Il 61820

Name __________
Address __________
City __________
State __________
Zip __________

Central data
P.O. Box 2484, Sta. A
Champaign, Il 61820
(217) 359-8010

Circle 45 on inquiry card.

SYNCHRO-SOUND

ENTERPRISES, INC.

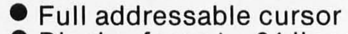
One-stop shopping for Hardware and Software

Everything you need in small computer systems with special emphasis on TERMINALS! Look at these units...compare price, quality, delivery, service...and you'll see why you don't have to look anyplace else!

LEAR SIEGLER ADM 3A TERMINAL

- Full addressable cursor
- Display format—24 lines of 80 characters per line
- Communications rates—75 to 19,200 Baud
- Computer interfaces —EIA standard

ADM 3A Kit........ **$759.95**
ADM 3A Assembled.. **854.95**
Lower Case option......... **69.00**

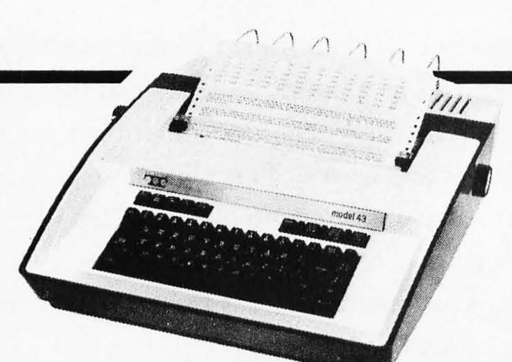


ADDS REGENT 100 VIDEO DISPLAY TERMINAL

Display Format—80 characters per line by 24 lines. 25th line reserved for terminal status. Reverse video, underline, half and zero intensity and blinking are all standard features.

$1325.00





TELETYPE MODEL 43 PRINTER

- 132 Characters per line
- 110 or 300 Baud switch selectable
- Full keyboard
- RS 232C Serial Interface standard

$1299.00

OKIDATA MODEL 110 LINE PRINTER

- 110 CPS dot matrix

Friction Feed... **$1199.00**
Tractor Feed..... **1349.00**
RS 232C Serial Interface... **260.00**



OKIDATA MODEL 22 LINE PRINTER

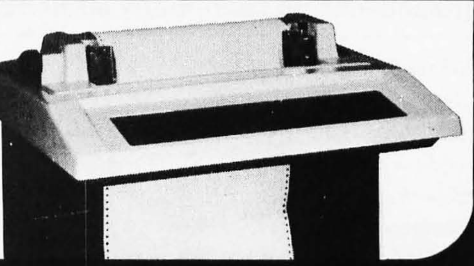
- 125 lines per minute 132-column print line
- Upper/lower case
- 8 different character sizes
- 12 IPS paper slew

Tractor Feed.... **$2449.00**
RS 232C Serial Interface... **379.00**

DECWRITER II

- 132 column printing
- 10-30 CPS
- Full keyboard
- Tractor feed

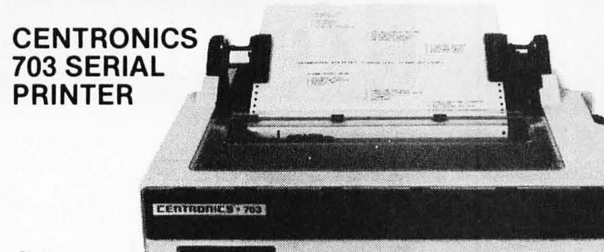
$1495.00



Circle 356 on inquiry card.

SYNCHRO-SOUND
ENTERPRISES, INC.

CENTRONICS 703 SERIAL PRINTER



- Low cost of ownership
- Bidirectional logic seeking printing
- Microprocessor electronics
- Excellent print quality

$2805.00

CENTRONICS 761 PRINTER
- 300 Baud serial transmission
- Bidirectional and incremental printing
- RS232, CCITT-V24, or current loop interface
- Baud selection (110/150/300)

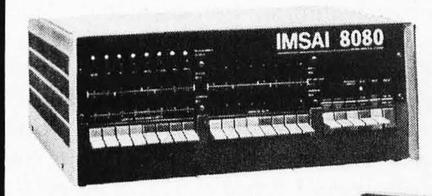
KSR with Keyboard....... **$2025.00**

Receive only version **1895.00**



IMSAI 8080 MICROCOMPUTER

● Powerful ● Low cost ● Easy to use

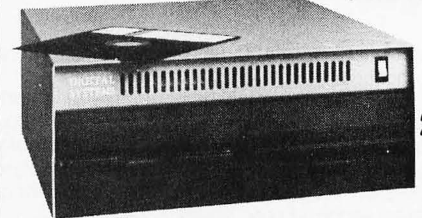


With 22 Slot Mother Board **$649.95**

PCS-80/15 Kit **699.00**

FLOPPY DISK SYSTEM
- Fully IBM 3740 media and format compatible
- Full formatter and controller built-in

Digital Systems Dual Drive System **$2475.00**



Dual Density Version **2745.00**

HAZELTINE 1500 VIDEO TERMINAL
- Reverse video
- 24 x 80 display
- Programmable brightness levels
- RS232 and current loop and much more

Assembled.... **$1149.00** /Kit... **$995.00**

HAZELTINE MODULAR 1 INTELLIGENT TERMINAL
- 1920 character display
- 8 different video levels
- Full editing capability
- Removable keyboard and much more

Assembled.... **$1659.00**



SPECIAL BUYS

Vista V80 Floppy Disk System	$619.95
Cromemco Z-2 Kit	565.25
Compucolor 8001 Color Computer	2595.00
IMS 16K Static Memory	525.00
North Star Microfloppy Disk Kit	599.00
North Star Horizon 1 Kit	1499.00
North Star Horizon 2 Kit	1899.00
Javelin 9″ Video Monitor	159.95
Livermore Modem Model 76	299.00
Micropolis Model 1053MOD 2	1799.00
IMSAI AP44-44 Col. Printer Kit	299.00
TDL Xitan Alpha 1 Assembled	939.95

We carry a full line of the following: TDL, Centronics, Seals, Hazeltine, Micropolis, Vista, Hayden, IMSAI, Cromemco, Digital Systems, Compucolor, Icom, Lear Siegler, Okidata, DEC, Javelin, North Star, Peripheral Vision. Same day delivery and shipping on most items. Full modern repair facilities on premises for complete servicing of everything we sell.



SYNCHRO-SOUND ENTERPRISES, INC.
The Computer People
193-25 Jamaica Avenue,
Jamaica, New York 11423
212/468-7067 TWX: 710-582-5886
Hours 9-4 daily and Saturday
Visit our new showroom
Working units on display
Dept. BBB BankAmericard • Master Charge

Languages Forum

A Homebrew Pascal Compiler

Herbert Stein
Sterzingerstrasse 18
8200 Rosenheim
Germany

Using Pascal as a language for systems programming at Fachhochschule München, my interest in Pascal grew to the point where I decided to write a compiler for myself. Begun last October, the syntax analysis stage, which was built up with recursive procedures, was written in two weeks. During the next few weeks I tested the program on a Cyber 175, having troubles with the original implemented Pascal compiler at first. On the 15th of November I received the first error free listing, but had to stop testing possible errors because some lectures and a computer graphics program had higher priorities. In the meantime, I worked out some of the next steps in theory.

During further expansion (which means space allocation, code generation and file handling), there are some potential difficulties:

- The MicroPascal compiler has to produce an optimized code, which allows real time applications and systems programs written in a high level language.
- The compiler needs features like garbage collection and dynamic space allocation for recursive subroutines or variable type declarations to keep the amount of runtime storage as small as possible.
- No existing monitor is able to run a language like Pascal efficiently. (The TDL system monitor board presented in the April 1978 BYTE seems to be headed in this direction.) So I will have to write a new operating system or transform an existing one to allow supervisor calls, IO interrupts, process handling and hardware interfaces for Teletype, video display and mathematical functions.

- The processor, which has to run the produced object code, should have bit instructions (like the Z-80), for handling set types. The Cyber 175 uses a 60 bit word for set types, which permits up to 59 elements in Boolean sets. The 8 bit words of microprocessors, which allow sets of up to eight elements, aren't sufficient for a compiler implementation. (I have had troubles during testing of the program on the Cyber 175 because I used a set type with 60 elements. I took a long time to discover my mistake after counting the set elements.)
- Should the compiler writer allow a GOTO in a language or not? If so, the user is able to leave a number of program blocks without being concerned about missing management routines, which are activated automatically at normal block ends.

  If the GOTO isn't allowed, the programmer has to write structured programs, using special instructions like *repeat . . . until* or *while . . . do*, and couldn't leave *begin . . . end* blocks arbitrarily. Each block is closed without additional program management for controlling unpredictable (at compile time) GOTO statements in object code.
- The last problem is that I don't own a microprocessor system, but intend to buy a small Z-80 system this month or next. Being familiar with instruction sets and operating systems of large machines, like Interdata, or larger ones, like the IBM 370, I have little experience with microprocessor systems. Some time will pass until I can build up and expand the microprocessor to be able to run cross-compiled Z-80 object code.

Working as a cross-compiler on the Cyber 175, my compiler will translate itself to a loadable form for a microprocessor system. If possible, I want to design another syntax analyzing stage. The compiler would pick up routines and expected sequence symbols from a table, which would contain the syntax description, depending on scanned input symbols. This technique would make error recovery much easier, because set types for sequence symbols would no longer be needed.

I hope to find people like Stephen P Smith, who are interested in an implementation of a Pascal compiler for microprocessors and who will inform each other personally or through BYTE's Languages Forum, because it seems impossible for individuals to tackle such a project.■

Clubs and Newsletters

Conducted by Laura Hanson

The Alliance OH Microcomputer Club

The Alliance Microcomputer Club is a recently formed organization located in Alliance OH. According to Gary S Fix, president, the group's goals are "to provide individuals in the Alliance area with the opportunity to share interests and experiences in exploring microcomputers as a hobby, career, social activity or curiosity." Meetings are held on the first Tuesday of each month at 7 PM. For further information about this new club, contact Gary at 3885 Norwood Av, Alliance OH 44601, or call him at (216) 823-8996.

SC/MP and SC/MP-II Users Group

Tom Bohon of Omaha NB has informed us of a SC/MP and SC/MP-II Users Group which has been formed. Members may take advantage of a library of both software and hardware information available on a cost basis. In addition, a bibliography of SC/MP articles, advertising, programming hints, etc, is available to members for the reproduction cost. The construction of a homebrew system based on the SC/MP-II is also in the planning.

No dues or fees are involved. However, in order to receive the monthly newsletter, send a self-addressed stamped envelope to Tom Bohon, 2215-A Walker Dr, Omaha NB 68123.

North Orange County Computer Club

According to Gary S Dickinson, the North Orange Computer Club is alive and well in Southern California. The correct mailing address is POB 3603, Orange CA 92655 and the phone number is (714) 998-8080.

KIM-1 Users Group

Anyone inerested in forming a KIM-1 Users Group in the San Fernando Valley area of California should write Jim Zuber, 20224 Cohasset #16, Canoga Park CA 91306, or call (213) 341-1610.

COSMAC-1802 Users Group

We have been notified by Patrick Kelly that a COSMAC-1802 Users Group is being formed for the purpose of corresponding, exchanging software and ideas and possibly publishing a newsletter. Membership is free and individuals

Continued on page 142

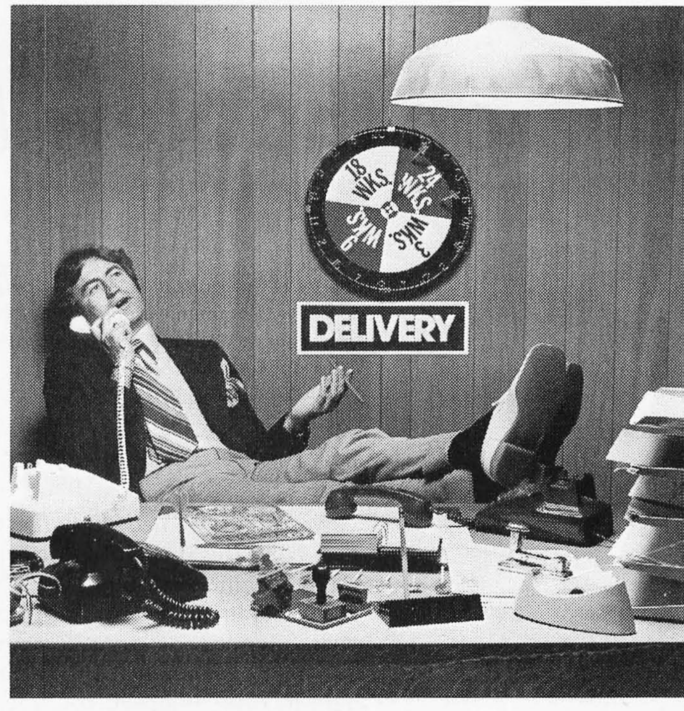
BUYING HARDWARE

Pros



Photo courtesy of ComputerLand of San Francisco

Cons



Computerland, the DP Department Store. Hazeltine & Diablo on Display.

Time was, buying hardware meant dealing factory direct. Then came the distributor, or factory indirect. The phone/mail order ad reads, "15% off." 15% off what? Delivery schedule maybe. After sales service perhaps. The only real savings just might be the Toll Free number in the ad.

Now there's an alternative. Introducing ComputerLand, a national network of DP equipment stores, with outlets across the nation. This means you, and the DP director, programmer, general manager or purchasing agent can take a quick ride across town, try out working equipment in a user environment, make product comparisons, then drive off with the hardware of your choice. And, still be back in time for the afternoon coffee break.

Below is a sampling of the quality business information products, from more than 60 manufacturers of micro and mini-computer systems and peripherals, available for you to try out, buy or lease and take home today from ComputerLand.

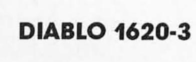


HAZELTINE 1500

☐ 24 X 80 Display (upper/lower case) ☐ High legibility 7 X 10 dot matrix ☐ Teletypewriter format for alphanumerics plus numeric pad ☐ Switch-selectable upper/ lower case ☐ R S -232/C 20 ma.current loop interface ☐ Switch-reversible video.

DIABLO 1620-3

☐ HyType daisy wheel character printer ☐ Full keyboard with numeric pad ☐ Microprocessor controlled ☐ R S -232/C interface ☐ 45 CPS ☐ Tractor feed ☐ Internal power supply.



ComputerLand™

14400 Catalina Street, San Leandro, CA 94577 (415) 895-9363

Franchise opportunities available.

ALABAMA

3020 University Dr., N.W.
Huntsville 205-539-1200

CALIFORNIA

6743 Dublin Blvd.
Dublin 415-828-8090

11074 San Pablo Avenue
El Cerrito 415-233-5010

22634 Foothill Blvd.
Hayward 415-538-8080

4546 El Camino Real
Los Altos 415-941-8154

6840 La Ciegega Blvd.
Inglewood 213-776-8080

24001 Via Fabricante #904
Mission Viejo 714-770-0131

289 East Highland Avenue
San Bernardino 714-886-6838

4233 Convoy Street
San Diego 714-560-9912

117 Fremont Street
San Francisco 415-546-1592

1077 Saratoga-Sunnyvale Road
San Jose 408-253-8080

42 - 42nd Avenue
San Mateo 415-572-8080

611 - 5th Street
Santa Rosa 707-528-1775

16720 So. Hawthorne Blvd.
Lawndale 213-371-7144

El Cid Plaza - Suite 104
Thousand Oaks 805-495-3554

104 W. First Street
Tustin 714-544-0542

1815 Ygnacio Valley Road
Walnut Creek 415-935-6502

6840 La Cienoga Blvd.
Inglewood 213-776-8080

COLORADO

2422 So. Colorado Blvd.
Denver 303 759-4685

CONNECTICUT

2475 Black Rock Turnpike
Fairfield 203-374-2227

DELAWARE

Astro Shopping Center
Newark 302-738-9556

GEORGIA

2423 Cobb Parkway
Smyrna 404-953-0406

ILLINOIS

50 East Rand Road
Arlington Heights 312-255-6488

9511 No. Milwaukee Avenue
Niles 312-967-1714

10935 S. Cicero Avenue
Oak Lawn 312-422-8080

KENTUCKY

813 B. Lyndon Lane
Louisville 502-425-8308

MARYLAND

16065 Frederick Road, (Rt. 355)
Rockville 301-948-7676

MICHIGAN

2927 - 28th Street, S. E.
Kentwood 616-942-2931

29673 Northwestern Highway
Southfield 313-356-8111

MINNESOTA

8070 Morgan Circle Drive
Bloomington 612-884-1474

NEW HAMPSHIRE

419 Amherst
Nashua 603-889-5238

NEW JERSEY

1442 E. Route 70
Cherry Hill 609-795-5900

2 De Hart Street
Morristown 201-539-4077

NEW YORK

1612 Niagara Falls Blvd.
Buffalo 716-836-6511

225 Elmira Road
Ithaca 607-277-4888

OHIO

1288 SOM Center Road
Mayfield Heights 216-461-1200

OREGON

12020 S. W. Main Street
Tigard 503-620-6170

TEXAS

3300 Anderson Lane
Austin 512-452-5701

6439 Westheimer
Houston 713-977-0909

WASHINGTON

14340 N. E. 20th
Bellevue 206-746-2070

1500 South 336th Street
Parkway Center, Suite 12
Federal Way 206-838-9363

8791 So. Tacoma Way
Tacoma 206-581-0388

WISCONSIN

690 S. Whitney Way
Madison (Directory Assistance)

INTERNATIONAL

52-58 Clarence Street
Sydney NSW 2000
Australia TLX: AA 24757

Circle 75 on inquiry card. "SEE US AT PERSONAL COMPUTING '78 IN PHILADELPHIA, AUGUST 24-27"

Compilation and Pascal on the New Microprocessors

Charles H Forsyth and Randall J Howard
Computer Communications Networks Group
University of Waterloo
Waterloo, Ontario
CANADA N2L 3G1

We are concerned with the use of high level languages, and in particular Pascal, on microcomputer systems. We are most interested in the use of such languages for what is termed, on larger computer systems, *systems programming*. This includes writing code to drive floppy disks, interpreters for APL or BASIC, or all those bits of code that people have until now written in assembler, and which in some way make their microcomputer systems friendly.

Microcomputer users show a generally high level of sophistication, so it might be surprising at first that so much of their code is still written in assembler. The advantages of writing in a high level language have been often described in computing literature: programs can be made more portable; they exhibit better structure; and they are easier to write and debug. In addition, it is much easier to let a compiler worry about the efficiency of the object code; and deficiencies of the object machine are hidden. With the 8 bit microcomputers like the Intel 8080 and Motorola 6800, we feel that there is little choice but to write in assembler (or interpreter), since the facilities provided by their order codes are simply insufficient to support most high level languages.

Compilation may be inappropriate for 8 bit microcomputers, but it is the most attractive alternative for the hybrid 8 and 16 bit microcomputers (such as the Motorola 6809), especially with respect to eliminating most assembly code on these machines. We also feel that Pascal has facilities that enable a compiler to generate better code for such machines than might be expected from compilers for other languages.

Jensen and Wirth provide the definition of and tutorial introduction to Pascal in the *Pascal User Manual and Report*. Aho and Ullman's book, *Principles of Compiler Construction,* provides an excellent description of the elements of a compiler.

Options

Tiny BASIC, Tiny C, APL, and FOCAL are implemented on microcomputers with interpretive code. Interpretation has a number of advantages. Since the interpretive language is highly specialized, it can be made compact. New *macro operations* can be added easily as time and experience dictate. Array and structure addressing and the block copying associated with array and structure assignment may be made particularly cheap. When interpreting array indexing, run time checks of the index values against the array bounds are possible (although often left out) at little extra cost. This is true of other kinds of debugging facilities as well, such as value traces or stack tracebacks. Both compiler and interpreter are easy to write, especially if the interpreted code implements a stack machine. Interpretation's main disadvantage is that it is slow.

An alternative to interpretation that alleviates this latter problem of speed somewhat is *threaded code,* which has been described as "interpretive code which needs no interpreter" (see references 2 and 3). Rather than having a sequence of codes and an interpreter which reads them, calling out to the routines implementing each operation, threaded code simply contains the sequence of machine addresses of the routines to process each operation. These routines, much like the code segments called by the inter-

```
type
        index = 0..10;
        twiceIndex = 0..20;
        unsigned = 0..32767;
        short = -128..127;
        shortUnsigned = 0..255;
        thing = record
                        field1: 0..7;
                        field2: 0..31
                end;
        packedThing = packed record
                        field1: 0..7;
                        field2: 0..31;
                end;
var
        a, b: array [index] of integer;
        i, j: index;
        k: twiceIndex;

        s: set of (READY, BLOCKED, RUNNING, SWAPIN, SWAPOUT);
begin
        a[i] := b[j];          {the dreaded array-indexing example}

        k := i+j;              {subranges are useful}

        s := [READY, BLOCKED, RUNNING];{set operations}
        s := s - [READY, RUNNING];
        s := s + [SWAPIN];
        s := s * [SWAPIN, BLOCKED];
end
```

Listing 1: Pascal program fragment for array indexing.

```
tsx                  /Enable indexing off sp
lda   A, j(X)        /Fetch address of j relative..
lda   B, j+1(X)      /to sp into (A,B) register pair
asl   B              /Shift (AB) pair left by 1..
rol   A              /yielding integer offset
add   B, b+1(X)      /Add in 16-bit array
adc   A, b(X)        /pointer i to (A,B) pair
sta   A, temp        /Transfer (A,B) pair to X reg..
sta   B, temp+1      /..not re-entrant
ldx   temp
lda   A, 0(X)        /Finally, fetch b[j] into..
lda   B, 1(X)        /(A,B) pair..
psh   A              /and push onto stack
psh   B
tsx                  /Following code is repeat of..
lda   A, i(X)        /above for getting address of..
lda   B, i+1(X)      /array element a[i]
asl   B
rol   A
add   B, a+1(X)
adc   A, a(X)
sta   A, temp
sta   B, temp+1
ldx   temp           /X now points at a[i]
pul   B              /Pop b[j] from stack..
pul   A              /into (A,B) pair..
sta   A, 0(X)        /and store in a[i]
sta   B, 1(X)

Total code: 52 bytes
```

Listing 2: Motorola 6800 assembly code for the first line of the Pascal fragment shown in listing 1.

The listings in this article were prepared by arrangement with Walter Banks of the University of Waterloo.

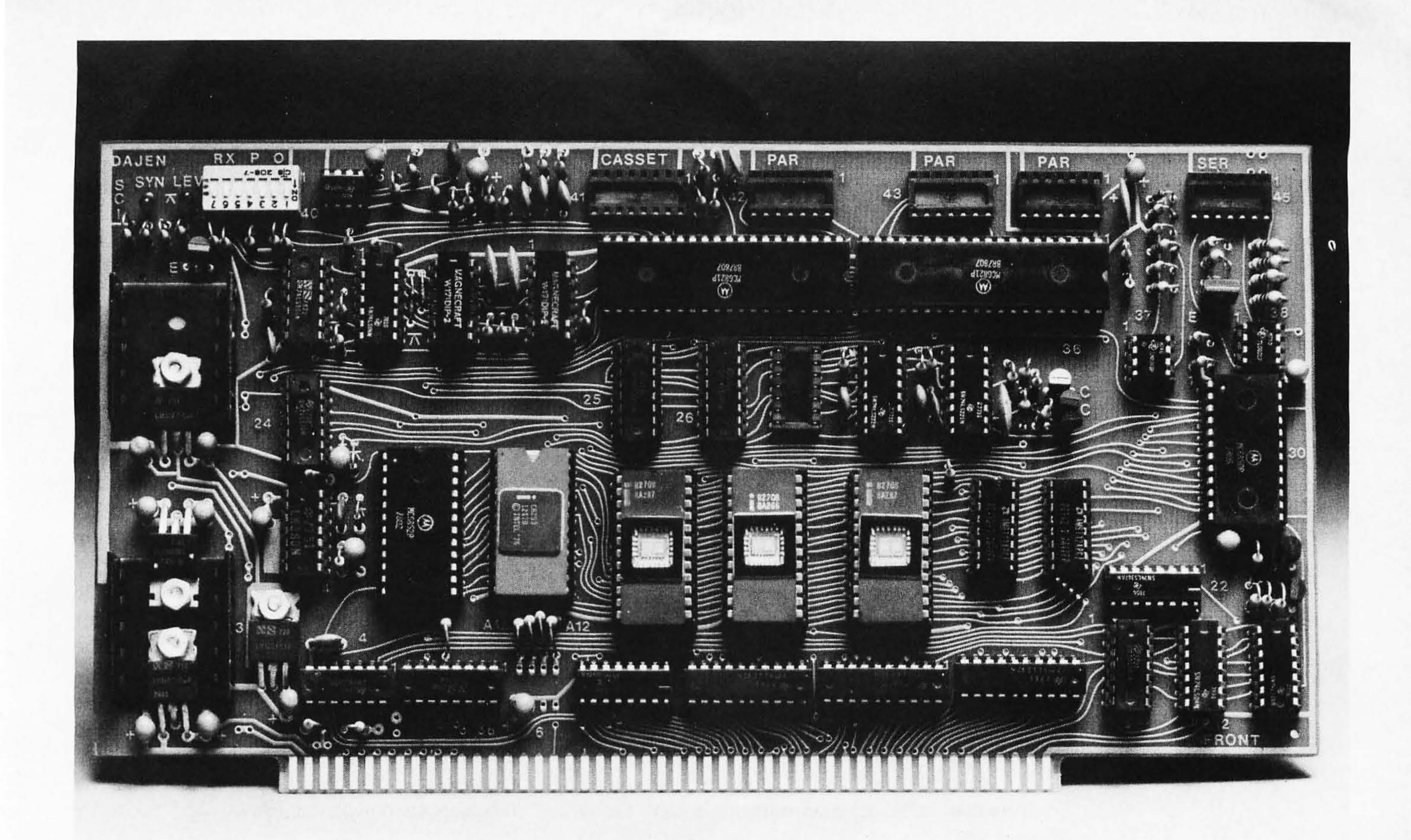
preter to implement the pseudo-machine, provide the run time support for the threaded code. Rather than return to an interpreter after it has done its work, though, a routine simply jumps (indirectly) to the next such routine in the code flow. Arguments are passed to these routines in various ways — for example, by placing values or addresses between the code pointers.

The third approach to language implementation is that traditionally adopted on larger machines: real code generation. This approach provides the fastest program execution at the possible expense of space used by the object code. On almost any machine, the high level constructs of flow of control and logical expressions as well as calls to the intrinsic built-in functions can be directly implemented as branch or jump instructions with relatively little expenditure of speed or time. However, for many of the existing microcomputers, code generation for even the simplest of the fundamental high level language constructs proves effectively impossible. Such constructs include most common arithmetic operations, array and structure accessing, and automatic storage manipulation. Particularly difficult on some machines are multiply, divide, modulus and string operations. Therefore it is important to determine what properties of a particular machine make it suitable for real code generation.

8 Bit Microcomputers

A detailed study of the common 8 bit computers available today (eg: Motorola 6800, Intel 8080) quickly reveals that such machines are not conducive to real code generation by compilers for high level languages such as Pascal.

On such machines, compilations of even the simplest arithmetic or pointer expressions lead to a very high object to source code ratio, if such constructs can be compiled at all. Listing 2 gives an example of code which might be compiled for a Motorola 6800 to implement the Pascal assignment statement: *a[i]:=b[j];* in listing 1. The assumption here is that automatic arrays are implemented as pointers on the stack to areas of storage residing elsewhere. In addition, we have assumed that the compiler keeps track of the stack offsets for its automatic variables relative to the moving stack pointer; we are using the notation *j* to represent the stack offset of variable j. In addition to this code segment, the procedure preamble must set up the pointers to the arrays a and b (stored at offsets *a* and *b* respectively), to point at the integer before the beginning of the array. Thus, for example, *a[1]* will then be identi-



The Most Powerful Interface.....Ever

Now for the first time all of these capabilities are available on one board; the System Central Interface (SCI).

— 2708 programmer and verifier; monitor provides necessary routines

— RAM stack: 256 bytes for stack and buffer

— Serial port: RS-232, 20mA, 60mA, programmable speeds from 45 to 9600 baud

— Three parallel ports, each with handshake signals

— Reset-jump (enables CPU to jump to the SCI Monitor)

— Biphase recording for highest speed, least error rate (to over 100,000 baud)

— Digital comparator — no adjustments ever

— S-100 compatable

Teletek provides quality interface systems backed by a reputation established over the last decade. Complete engineering support services are available for OEM applications.

For more information call Ann Roberts at (916) 351-0535 or write us and we will send complete information.

DEALER INQUIRIES INVITED
OEM CONTRACTS AVAILABLE

TELETEK ENTERPRISES INC.

11505B Douglas Road
Rancho Cordova, CA 95670

Circle 371 on inquiry card.

Let	
	r := { X, Y, S, U }
	a := { A, B, D }
	x := memory reference
	c := constant value
x	long relative, short relative, direct
$*x$	long & short relative indirect
$\$x$	immediate byte
$*\$x$	extended
$**\$x$	extended indirect
$c(r)$	±4, ±7, ±15 bit indexing
$*c(r)$	±7 and ±15 bit indirect indexing
$(r)+$	Auto Increment by 1 or 2
$-(r)$	Auto Decrement by 1 or 2
$*(r)+$	Indirect Auto Increment by 2
$*-(r)$	Indirect Auto Decrement by 2
$a(r)$	Accumulator Indexing
$*a(r)$	Indirect Accumulator Indexing

Table 1: A summary of the Motorola MC6809 addressing modes.

fied with the beginning of the storage associated with the array a.

Beyond the actual code shown here, however, the most important insight to be gained from all of this is the sheer bulk of code that such a simple construct would generate (and it is not even reentrant at that). Imagine how large the object code size would be for even a reasonably short Pascal program.

Implementing threaded code is somewhat difficult on these machines because they require 16 bit memory pointers, an efficient mechanism for indirect addressing, and some method of incrementing such a pointer to the next 16 bit pointer. At least one of the above criteria is so troublesome on both the Motorola 6800 and the Intel 8080 that the threaded code becomes unwieldy. Thus, for these machines one has little choice but to interpret or write in assembler. This suggests that the interpreters themselves must be implemented in assembly language.

The above discussion is an attempt to analyze the reasons why programs written for 8 bit microcomputers have traditionally been interpreted or written in assembly or machine code, rather than being compiled into "true" code from a high level language.

16 Bit Microcomputers

Previously, the only alternative to the 8 bit architecture was that of the 16 bit microcomputer. Examples of such machines include the TI-990/4 and the DEC LSI-11. While the considerable costs of these processors tend to make them impractical for many computer experimenters, and for those applications in which many processors are required, it is instructive to consider what properties set these machines apart from their 8 bit counterparts with respect to code generation. In fact, it can be shown that, given a machine of sufficient sophistication, it should be possible for a compiler to do as good a job as an assembler programmer vis-à-vis machine resource utilization.

There are two main virtues of these 16 bit machines. In the first place, these machines have complete 16 bit instruction repertoires including hardware multiplication, division, and long shifts. As well, the 16 bit processors tend to have a good complement of addressing modes such as indexing, stack operations, automatic increment and decrement of pointers, and so on. (Here, as elsewhere in this article, the descriptive terms may seem fuzzy. *Good complement* does not admit of a precise meaning. With real machines, one usually loses clever addressing modes, for plenty of general purpose registers, and one must balance the benefits somehow. The final judgment will usually be that of the person writing the compiler.) With these attributes, it is a fairly straightforward task to construct a compiler for a high level language such as Pascal.

8 and 16 Bit Hybrids

The current trend in 8 bit microprocessor technology is towards a hybrid combination 8 and 16 bit machine. Essentially, these processors are capable of 16 bit operations while retaining 8 bit data paths throughout the processor architecture. A prime example of such a hybrid is the Motorola 6809, which is due for formal product release later this year. Table 1 gives a summary of the basic addressing capabilities of the Motorola 6809, expressed in a hypothetical assembler syntax which removes from the user the burden of understanding all of the details of the actual hardware addressing modes.

What advantages do these machines have over their pure 8 bit predecessors? In particular, these machines now have at least one accumulator for performing addition, subtraction, shifting and comparison operations on 16 bit data. A second feature of these machines is the 16 bit memory pointer, which, combined with the ability to automatically increment and decrement such pointers, provides a very general memory accessing capability. In addition, common high level language features such as stack frames and display pointers become quite easy with the general index and stack registers of the M6809. It is apparent that the Motorola 6809 is particularly well-endowed with addressing modes which tend to facilitate code generation for high level languages.

Home-Computer Retailers Get a new Sales Tool With



Software

READY-TO-USE PRS PROGRAMS ENCOURAGE HOME-COMPUTER SALES.

Enable retailers to offer home-computer buyers an easy road to computer usage. An answer to the doubters who question the use of a computer in the home. These are thorough and exciting applications. The elimination of the old "tic-tac-toe" program.

NEW STANDARD OF EXCELLENCE IN DOCUMENTATION SIMPLIFIES RETAILER'S AND USER'S LIFE.

Retailers, you can now realize your dream .to put a computer in every household. Each program of PRS Corporation comes with complete written documentation in clear and easy-to-understand English. And avoids esoteric technical language. Indeed, retailers, you can now offer homeowners a fluency at their fingertips without their having a Ph.D. in Computer Sciences.

DISPLAY RACK SPARKS COMPUTER SALES.

Enables home-computer buyers to browse through beautifully packaged cassette-folders. Each prospect can study various types of programs and become interested in the advantages of computers.

EXTENSIVE PROGRAM CAPABILITIES MAKE COMPUTERS DESIRABLE.

With "microfile," a new and revolutionary datafile management program, users can:

- Manage any list of items
- Enter new data
- Edit information already on file
- Order items alphanumerically upwards and downwards
- Sort by primary and secondary keys
- Search out an individual item or print out entire list
- Total sums of columns
- Justify right and left margins of each column
- Delete items
- Save and retrieve data with mass storage media

PROGRAM APPLICATIONS EXPAND COMPUTER USAGE:

Gives retailers opportunities to tell home-computer prospects about the unending list of home and personal usages. For example, microfile manages:

Home inventory of furnishings • Listing of musical tapes • Calendar of activities • Vacation itineraries • Checks by categories • Home-upkeep tickler file • Family historical dates • Health records

3 LEVELS OF PROGRAM-USAGE.

Educational and games; homeowner and business applications; sophisticated monitors and programming tools.

PROGRAMS AVAILABLE FOR THE MAJOR COMPUTER BRANDS:

1. SOL from Processor Technology Corp. and all computers with Solos/Cuter monitor
2. TRS-80 from Radio Shack
3. The Apple II from Apple Computer Inc.
4. The Pet Computer from Commodore
5. The Sorcerer from Exidy Inc.
6. Standard set adaptable to other systems with Z-80, 8080A, 6502 microprocessors

EXCLUSIVE SALES TO RETAILER,

PRS Corporation offers only retailers the sales opportunity for PRS software. With no direct sales to end users. Each retailer is provided with a beautiful informative rack capable of holding up to 20 cassette-folders. Microfile (a Level 2 program) and DDS II (a Level 3 program) the famous Dynamic Debugging System—are now available with additional programs coming on a regular schedule.

CALL NOW FOR YOUR SPECIAL TRIAL DISCOUNT OFFER:

Call immediately (212) 787-1526 or 787-1527; or write today for details to: PRS—THE PROGRAM OF THE MONTH Corporation, 257 Central Park West, New York, N.Y. 10024.

Circle 306 on inquiry card.

Consider again the array assignment which the 6800 handled so dismally. The Motorola 6809 code for the same construct is given in listing 3. (Note that the syntax of our assembler code is intended to be more or less consistent amongst the examples, and not necessarily that of the manufacturer's assembler. It is in fact the syntax used by our UNIX assemblers for these machines.) Code for the PDP-11/45, considered to be a good instruction set given in listing 4, is included for comparison.

It is rather precipitous to deduce much from this one example, although array indexing does exercise many of the addressing modes of a machine, and such assignment statements can provide a check on the register usage of a compiler. How a particular architecture fares with more general arithmetic expressions and function and procedure call, save, and return sequences would provide further basis of comparison. Indeed, other examples that we have tried suggest that the results of this comparison are typical.

Special Advantages of Pascal

We feel that the use of Pascal and a competent compiler can lead to better code in many cases on hybrid 8 and 16 bit machines than can be achieved with many other languages. Obviously, the best results will require that Pascal be properly used — that subranges be used where possible, for example — and that these be declared to be as small as possible. A Pascal program can contain a great deal of information that allows even a straightforward compiler to generate code which makes good use of the available registers. The Pascal declarations of listing 1 provide illustration for the following discussion, and the code given is for the Motorola 6809. Remember that the intent is not to describe an implementation of Pascal.

The declaration of scalar and subrange types essentially allows the declaration of *small* integers and makes known the detailed characteristics of variables of such types to the compiler. Variables may thus be completely bounded, and the compiler can compute upper and lower bounds on the value of an expression.

In our example, variables of type *short* or *shortUnsigned* may be loaded into the 8 bit accumulators of the 6809, and both registers may be used simultaneously. A variable may be recognized as *unsigned* if there are no negative values in the subrange to which it belongs. In the assignment statement *k := i+j;* the variables *i*, and *j*, are both in the range 0 thru 10. The result is thus in the range 0 thru 20, and an 8 bit accumulator may again be used to compute this result. (All of this is particularly useful if array indexing is also involved.)

The Pascal *set* type may be regarded as providing a readable way to do "bit twiddling." A set is typically implemented as a sequence of bits, one for each element of the base type of the set. The variable *s* might then be a byte in which the low order bit corresponds to the element READY, the next to BLOCKED, and so on. The sequence of assignments might then be compiled as in listing 5.

Pascal, of course, provides pointers, **record** structures and arrays.

The use of pointers is strictly controlled: arbitrary arithmetic operations on pointers are not allowed. About the only things that may be done with a pointer variable are: indirect addressing, assigning another pointer to it, or passing it to a procedure or function. This structured use of pointers and indexing results in a very stylized use of pointers in the compiler's internal representation. This in turn allows the compiler to detect the places where double indexing may be used to advantage rather easily, on machines like the 6809 which have this feature.

Indexing of an array of records does require multiplication of the index by the width, in bytes, of the record. Often, this may be accomplished by a shift. Of course, this cannot always be done, since records need not be a power of 2 in length, though a compiler could arrange to round the size of a record up to an appropriate boundary if

```
/ 'X' points to top of stack (display)
lda     D, i(X)     / i
asl     B
rol     A           / *2
add     D, $a-2     / +offset of 'a'
lea     Y, D(X)     / +stack top
lda     D, j(X)     /j
asl     B
rol     A           / *2
add     D, $b-2     / +offset of 'b'
lda     D, D(X)     / +stack top
sta     D, (Y)      / a[i] := b[j]

Total code: 20 bytes
```

Listing 3: Motorola MC6809 assembly code for array indexing program fragment.

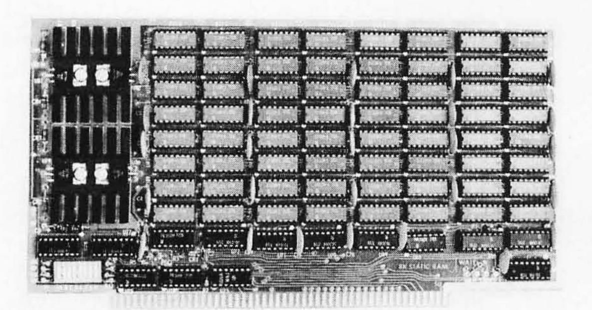
```
/ r5 points to the "top" of the
/ stack frame
mov     j(r5),r0        /j
asl     r0              / *2
add     r5,r0           /+ display pointer
mov     i(r5),r1        /i
asl     r1              / *2
add     r5,r1           /+ display pointer
mov     b-2(r0),a-2(r1)         / a[i] := b[j];

Total code: 22 bytes
```

Listing 4: DEC PDP-11 assembly code for array indexing example.

Wondering which memory is best for you?

base 2• offers the following products to the S-100 market at the industry's lowest prices:

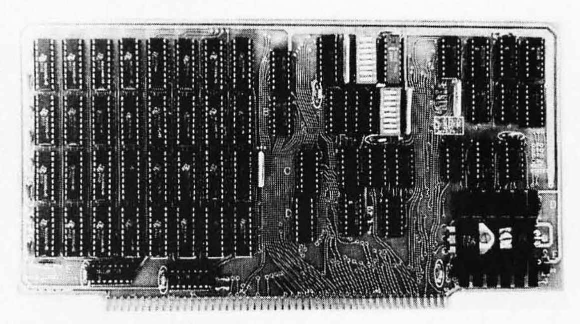


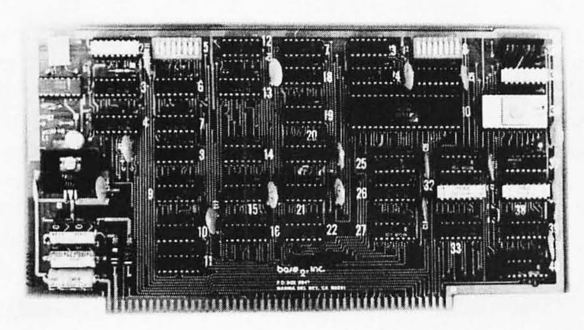
8K Static Memory Board

This 8K board is available in two versions. The 8KS-B operates at 450ns for use with 8080 and 8080A microprocessor systems and Z-80 systems operating at 2MHz. The 8KS-Z operates at 250ns and is suitable for use with Z-80 systems operating at 4MHz. Both kits feature factory fresh 2102's (low power on 8KS-B) and includes sockets for all IC's. Support logic is low power Schottky to minimize power consumption. Address and data lines are fully buffered and 4K bank addressing is DIP switch selectable. Memory Protect/Unprotect, selectable wait states and battery backup are also designed into the board. Circuit boards are solder masked and silk-screened for ease of construction. These kits are the best memory value on the market! Available from stock . . . **8KS-B $125** (assembled and tested add $25.00)
8KS-Z $145 (assembled and tested add $25.00)

16K Static Memory Board

Base 2 can now offer the same price/performance in a 16K static RAM as in its popular 8K RAM. This kit includes 8K bank addressing with 4K boundary address setting on DIP switches. This low power unit provides on-board bank selection for unlimited expansion . . . No MUX board required. Using highest quality boards and components we expect this kit to be one of the most popular units on the market. Available in two speed ranges, the 16KS-B operates at 450ns while the 16KS-Z operates at 250ns.
16KS-B $285 (assembled and tested add $25.00)
16KS-Z $325 (assembled and tested add $25.00)



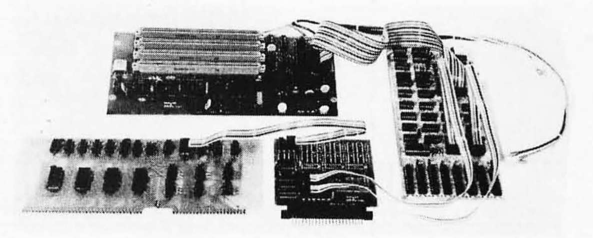


Z-80 CPU Board

Our Z-80 card is also offered in two speed ranges. The CPZ-1 operates at 2MHz and the CPZ-2 operates at 4MHz. These cards offer the maximum in versatility at unbelievably low cost. A socket is included on the board for a 2708 EPROM which is addressable to any 4K boundary above 32K. The power-on jump feature can be selected to address any 4K boundary above 32K or the on-board 2708. An On-board run-stop flip-flop and optional generation of Memory Write allows the board to run with or without a front panel. The board can be selected to run in either the 8080 mode, to take advantage of existing software, or in the Z-80 mode for maximum efficiency. For use in existing systems, a wait state may be added to the M1 cycle, Memory request cycle, on-board ROM cycle, input cycle and output cycle. DMA grant tri-states all signals from the processor board. All this and more on top quality PC boards, fully socketed with fresh IC's. **CPZ-1 $110 CPZ-2 $125**

S-100 for Digital Group Systems

This kit offers, at long last, the ability to take advantage of S-100 products within your existing Digital Group mainframe. Once installed, up to four S-100 boards can be used in addition to the existing boards in the D.G. system. The system includes an "intelligent" mother board, ribbon cables to link existing D.G. CPU to the DGS-100 board and a power wiring harness. The DGS-100 is designed to fit in the 5-3/4" x 12" empty area in the standard D.G. cabinet. It may seem expensive but there's a lot here! End your frustration! **DGS-100 $295**



base 2• inc.

Send for more details on these products. Get on our mailing list for information on more soon to be announced products at factory-direct prices from BASE 2. Why pay more when you can get the best at these prices???

P.O. Box 3548 • Fullerton, Calif. 92634
(714) 992-4344
CA residents add 6% tax
MC/BAC accepted • FOB — U.S. destination

Circle 19 on inquiry card.

```
/ X is display pointer
/ equates are in octal
READY = 01
BLOCKED = 02
RUNNING = 04
SWAPIN = 010
SWAPOUT = 020
lda     A, $READY+BLOCKED+RUNNING / immediate load
sta     A, s(X)
/
lda     A, s(X)
anda    A, $![READY+RUNNING] / complement
sta     A, s(X)
/
lda     A, s(X)
ora     A, $SWAPIN
sta     A, s(X)
/
lda     A, s(X)
anda    A, $[SWAPIN+BLOCKED]
sta     A, s(X)
```

Listing 5: Set assignment code for the Motorola MC6809 processor.

the difference were small. In any event, provided the size of the record is no more than eight bits (as an unsigned quantity), the code for the multiplication could reasonably be included in line.

We wondered how often division or multiplication is used in the UNIX system (an operating system developed at Bell Labs), and wrote a simple command file which would compile each of the source programs of the system and scan the resulting assembler for *mul* and *div* instructions. The number of multiplications was of interest in light of the above discussion; the number of divisions was collected as well, since these would have to be interpreted by subroutine on the 6809, and we wanted to know how many occurred in critical code. The results are shown in table 2.

Only one of the divide instructions occurs in a routine that might be regarded as significant, with respect to increasing system overhead, were a subroutine called to do the divide piecemeal; and that division was performed at a low priority level. 31 of the divide instructions in the device driver routines were in disk drivers, which had to compute track and cylinder offsets. The multiplications in all cases were of small amounts; it seems that (most likely by accident) record structures used in the kernel happened to be a power of 2 in length. It would have been more instructive, perhaps, to examine user programs, but in that case it would have been more difficult to separate multiplications written explicitly from those created implicitly by array indexing.

Section	Lines of C Code	Number of Multiplications	Number of Divisions
UNIX Kernel	6,013	4	9
Device Drivers	8,640	62	41

Table 2: A search through a particular operating system to determine the number of multiplications and divisions used. This was done to determine how important the speed of a multiplication and division routine would be to a typical program.

A Pascal programmer may declare particular record or array types as *packed*, which is a hint to the compiler that the programmer would prefer elements of the given type to occupy as little space as possible even if there is a cost in increased code to access them. This leaves the unit of packing to the compiler. For example, the types *thing* and *packedThing* (see listing 1) describe packed and unpacked records with similar fields (to Pascal, these record types are not compatible in any way). In a *thing*, both *field1* and *field2* will likely be bytes, but if a compiler implements the notion *packed* completely, then in a *packedThing*, *field1* will likely occupy three bits, and *field2* five bits, ie: they would share the same byte of storage. Packing of records on microcomputers is often much easier than on the larger processors, because microprocessors do not have the alignment problems that plague compiler writers on those machines.

Finally, as in many other languages, the order of evaluation of expressions is left to the implementor, but since side effects are not allowed, no legal Pascal program can possibly be harmed by this. This has two related effects: in arithmetic expressions, the compiler may evaluate the operands in the order that leads to the least amount of code, and in Boolean expressions the left-hand side of the logical operators *and* and *or* need not be evaluated if the expressions on the right determines the truth value of the entire expression. Faster or smaller code will usually result if a compiler takes advantage of these properties.

Pascal: Problems?

We feel that there are a number of areas where Pascal is likely to require expensive mechanisms, and which would be inappropriate for a systems programming environment. One solution might be to implement a subset of the language, leaving these hard features aside, but in most cases, since the expensive mechanisms are only invoked if the programmer asks for them, it should be sufficient to have the compiler avoid including the associated run time procedures when they are not requested. (This is worth mentioning, if only because this rule is often not followed.) We shall first mention those

constructs which are expensive, but which appear only by programmer request.

The semantics of Pascal's *file* variables, and the input/output (IO) system in general tend to reflect characteristics of a batch environment, with a restricted character set. The basic IO procedures are badly designed for an interactive terminal. The *read* and *write* procedures are fairly expensive to implement, since they are extremely general and all encompassing.

On machines like the 6809 which lack a divide instruction of any sort (let alone a 16 bit one), division will be done by calling a run time support routine. Only if the programmer explicitly writes either a divide, or modulus operation, will the call be generated. Floating point numbers will be interpreted, as usual.

Pascal allows procedures and functions to be defined inside other procedures and functions. This requires either a display, which must be copied, or a system of pointers by which a routine may access the variables owned by routines in an outer scope. (The latter is the most likely choice.)

Strings, arrays, records and large sets (if implemented) may all be assigned or passed as parameters to routines. These operations require block copies, but only if the operations appear in the source program. Copying of actual parameters may be avoided, of course, by declaring the matching formal parameters as *var* parameters.

The remaining points concern some philosophical concerns about Pascal and its implementation. (Input and output might also be considered in this class.)

Philosophy

It has been observed that much of the checking done at run time in other languages may be done at compile time in Pascal. This is not always so, and run time checks are required on assignments of a variable from a larger subrange to a variable in a smaller subrange of a given type, or on similar use in array indexing, and pointers must always be checked to ensure that they are not *nil*. It might be argued that run time checks might not be done at all. It is better to arrange for them to be turned on and off, as required, in different sections of code.

The *Pascal Report* (see references) does not put boundaries on the number of elements in the base type of a *set* type, but it does say that an implementor will likely choose the word length of a given computer as that limit. Otherwise, routines are required to perform various Boolean operations on large bit strings. Unfortunately, a great many Pascal programs in existence, most notably those for the CDC 6600, assume that it is possible to delcare or use a *set of char*, as in:

```
if c in ['a'..'z'] then
    { c is a letter }
```

where *c* is declared as a *char*. The CDC Pascal compiler restricts the number of elements in the base type of a set to about the number of bits in a word (58), but the CDC character set is small enough that it (nearly) fits within a set. On a microcomputer with the ASCII character set even 16 bits is clearly insufficient, and larger sets may need to be implemented.

There is no method provided to initialize variables in their declaration. This is of consequence when one wishes to create a table with values that remain constant throughout the life of the program (eg: a translation table). The only way to do this in standard Pascal is to write a sequence of assignment statements. This will typically result in several bytes of code for each assignment, as well as forcing two copies of each data value in the table. On a large machine like the CDC 6600, this may be of

little consequence, but on a microcomputer with little core, this is a distinct disadvantage. Of course, various implementations of Pascal have provided a means to do this sort of thing efficiently, but this results in a portability problem because each implementor tends to have slightly different rules about where and how these initializations may be accomplished.

Conclusions

For languages like Pascal, compilation is the preferred method of implementation on hybrid 8 and 16 bit microprocessors. The object code size on these machines for common constructs in these languages seems to compare quite favorably with that for larger processors like the PDP-11 or the Honeywell 66/60. We illustrated this with a very simple array operation; the reader can try other operations.

When choosing a programming language, one typically considers not only the ease or difficulty of implementation and the efficiency of the compiled code, but stylistic qualities as well. For example, we have found the C language a pleasant and effective language for developing programs, but it does not, of course, follow that everyone else would. The same holds true for Pascal. We merely note that the Pascal is interesting, in that Pascal programs may be so written as to allow a compiler to compile code which makes efficient use of 8 bit accumulators on machines that have them, and that amongst the other major high level languages this is an unusual property (PL/I is a likely exception). Whatever the language used, we hope to see the day when on microcomputer systems, as on UNIX, the use of assembly language for a program of any size is greeted with surprise, shock, depairs, dismay and outright hostility.■

REFERENCES

1. Jensen, K and Wirth, N, *Pascal User Manual and Report*, Springer-Verlag, New York 1975.
2. Bell, JR, "Threaded Code," *CACM*, volume 16, number 6, June 1973, pages 370 thru 372.
3. Dewer, RBK, "Indirect Threaded Code," *CACM*, volume 18, number 6, June 1975, pages 330 thru 331.
4. Aho, AV, and Ullman, JD, *Principles of Compiler Construction*, Addison-Wesley, Don Mills, Ontario 1977.
5. *990 Computer Family Systems Handbook*, manual number 945250-9701, Texas Instruments, Austin TX 1976.
6. *LSI11 PDP11/03 Processor Handbook*, Digital Equipment Corp, Maynard MA 1975.
7. *M6800 Microprocessor Programming Manual*, Motorola Semiconductor Products, Phoenix AZ 1975.
8. Kernighan, BW, and Ritchie, DM, *The C Proprogramming Language*, Prentice-Hall, Englewood Cliffs NJ 1978.
9. Thompson, KL, and Ritchie, DM, "The UNIX Time-Sharing System," *CACM*, volume 17, number 7, July 1974, pages 365 thru 375.
10. *Honeywell 66/60 Macro Assembly Program*, Honeywell Information Systems, Phoenix AZ 1972.
11. Wiles, et al, "Compatibility Cures Growing Pains of Microcomputer Family," *Electronics*, 2 February 1978.
12. *M6809 Advanced Microprocessor*, Motorola, Austin TX.

BYTE's Bugs

The Price Is Wrong

In the book review of *The Elements of Programming Style,* which appeared in May 1978 BYTE, page 161, the price of the book should have been $5.95, not $2.65. Thanks go to P J Plauger for notifying us of this error.■

Transposition Bug

We apologize for any inconvenience to readers' internal interpreters caused by the slight deviation in scanning between pages 164 and 165 in the June 1978 issue. The two columns of text on page 165 were inadvertently reversed. Transposition of these two columns will restore the correct syntactic order of the text (see below).■

Still Further Thoughts

On page 122 of the June 1978 BYTE there appeared a Language Forum item called "An APL Interpreter: Further Thoughts." In paragraph 2 Tom Brightman remarks that most reductions are monadic. David Eisenstein contacted us to state that most reductions are dyadic and research into several APL books verified this.

I feel that the problem here is one of interpretation. If a reduction function, such as +/, is considered to be one operation, then the operation is monadic. The one operand is to the right of the function and is usually a vector.

However, the reduction function is usually defined to be only the left slash symbol. This means that the reduction function is dyadic. The righthand operand would usually be an array and the lefthand operand would be some operator such as + or x. This is the form that texts seem to use. . .RGAC■

System Clear

When a CTRL/W followed by a Clear is detected in the Control Check Section, a jump is made to INITialize and all system parameters are initialized. The screen is cleared and the cursor is moved to the upper left-hand corner.

System Clear is designed to be accessed by pressing CTRL/W and then Clear, to avoid accidental use. It is, however, occasionally handy to be able to reinitialize the entire program.

Clear Screen and Home Cursor

When only the Clear key is pressed, the program jumps to CLEAR. This clears the screen and returns the cursor to the upper left-hand corner. Memory status words are unchanged.

Home Cursor

When a CTRL/X is detected in the Control Check section, the program jumps to Home, which returns the cursor to the upper left-hand corner but does not affect the screen contents or the memory status words.

Escape

When the escape key (ESC) is pressed, it is detected by the Control Check section and a jump is made to hexadecimal memory address 345 where the program receives instructions for exit from GRAPH (see Program Function and Use).

Addressing and Memory Requirements

In its present assembly, GRAPH resides in hexadecimal memory locations 000 thru 3FFF and is designed to drive a VDM-1 addressed at hexadecimal CC00. VDM-1 status port (to reset scrolling) is addressed at hexadecimal C8.

In addition, six 1 K byte memory sectors are set aside for the STORE and RECALL functions (see table 3).

A keyboard inputs status information to IO port 00 (data present = bit 6 set) and data to IO port 01.■

A 300 bps cassette of GRAPH in Kansas City, BYTE or Cuter formats with a CUTER header is available for $5 (cash or money order) from UNB Audio Visual Services, UNB, Fredericton NB CANADA E3B 5A3.

END

Sector	Hexadecimal Beginning Address	Hexadecimal Locations of Begin Address in GRAPH Program	Hexadecimal End Address Plus One	Hexadecimal Location of End Address in GRAPH Program
1	0800	22F, 249 (08)	0C00	24B (0C)
2	0C00	234, 25B (0C)	1000	25D (10)
3	1000	239, 263 (10)	1400	265 (14)
4	1400	23E, 26B (14)	1800	26D (18)
5		273 (18)	1C00	275 (1C)
6		C)	2000	270 (20)

Table 3: Six 1 K byte memory sectors which are set aside for the STORE and RECALL functions.

START

if the program is in the vertical write mode). It then adds the value 80 to the accumulator and deposits the result back into 3F7. This has the effect of alternately loading that location with hexadecimal 00 or 80 every time this routine is entered. Thus, one stroke of the appropriate key (CTRL/Q) puts you in the Vertical Write mode, and another stroke takes you out of that mode. Operation then jumps back to STAT!N in the Driver.

Cursor (On/Off)

This operation works exactly the same as Vertical Write by alternately loading hexadecimal location 3FB with hexadecimal 00 or 80.

Cursor Write/Don't Write

Cursor write/don't write works exactly like Vertical Write, alternately loading location hexadecimal 3F6 with hexadecimal 00 or 80.

Next Store

When a CTRL/W and a number from 1 to 6 are detected (see Control Check Section), the Next Store routine is entered. This routine is actually only a series of comparisons in the Control Check Section which compare the input character with several ASCII hexadecimal values. For example, after CTRL/W and a numeral 1 are entered from the keyboard, the data input to the computer is hexadecimal 31. In this case a CPI 31 instruction would route the program to instructions that would load status word hexadecimal memory location 3F8 with hexadecimal 00, and when 3F8 is checked by the next STORE operation, it would store page 1 in the first memory sector. Entering a 3 would be detected by a CPI 33, which would load 3F8 with hexadecimal 02 and set up the next STORE operation for page 3.

June 1978 BYTE Publications Inc 165



Finally, the macro maker of computer printers has gone micro. With a high-speed, desk-top, non-impact printer complete with parallel interface (Model P1) that's perfect for the home, hobby and small business user. A complete 80-column printer for less than ~~$600~~.

The new Centronics microprinter is big on performance. It prints clean, clear dot matrix characters at a rate of 150 lines per minute. Yet it's quiet, compact, comes fully warranted and is backed by Centronics' worldwide service. It's even available in a serial interface version (Model S1) for remote applications.

Of course, buying your small systems printer from the industry's leading producer of printers is simply a better buy. You get the security, support and systems flexibility only Centronics can bring. Plus the targeted technology that brings out the best in any computer.

So when you think small printers, think big. Think Centronics. For more information, just clip and mail the coupon below.

Centronics Data Computer Corp.
Hudson, NH 03051

Gentlemen: Tell me more about your new microprinters.

Name ____________
Title ____________
Company ____________
Address ____________
Town ____________
State ____________ Zip ____________
Phone ____________

B878

CENTRONICS® PRINTERS
Simply Better

The Number Crunching Processor

Peter Nelson
68 Seaver St
Wellesley MA 02181

National Semiconductor Corporation's MM57109 microprocessor is designed specifically for numeric processing. Called "The Number Cruncher" in their advertising, the MM57109 has an instruction set that includes floating decimal arithmetic, logarithmic and trigonometric functions and other sophisticated features. Although it can be used as a stand alone device with read only and programmable memory, or as the "brain" of a smart instrument, most hobbyists will probably want to use it as a peripheral processor where it will save both money and memory space.

The MM57109 requires a 9 V power supply which can be configured as +5 V and −4 V for easier TTL interface. It also requires a single phase 5 V clock of about a

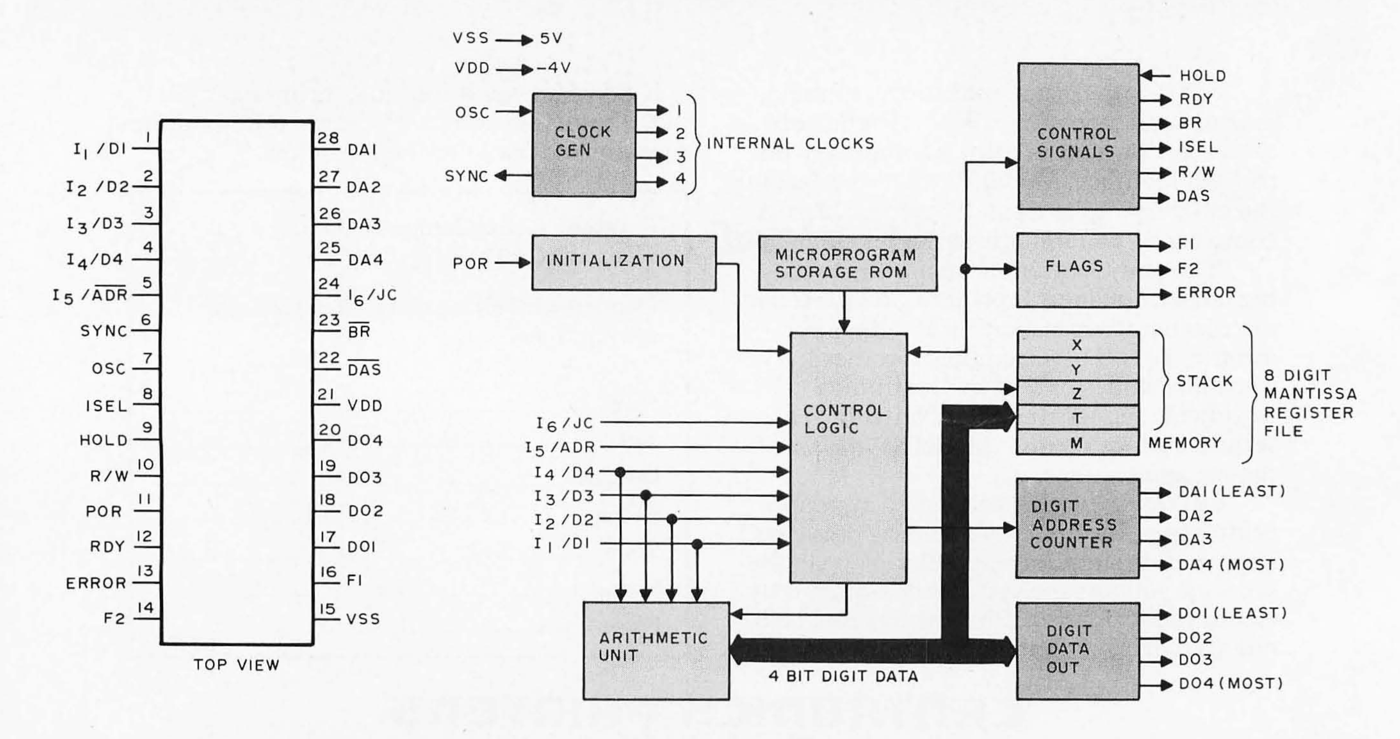


Figure 1: Pin assigments for the dual in line packaged MM5701 and a functional block diagram of the processor. Reproduced courtesy of National Semiconductor Corporation.

400 kHz frequency. This is internally divided down to a 100 kHz SYNC signal which forms the basic time period of the processor. National calls this 10 μs period a microcycle. All instruction times are expressed in microcycles.

A pinout and block diagram are shown in figure 1. The MM57109 receives instructions and data via input lines I_1 thru I_6. Timing of an instruction or data fetch operation is shown in figure 2. Note that the RDY output goes high to signal that input is required. I_1 thru I_6 may change only when the RDY line is high. Processing begins when it returns to a logic "0," 8 microcycles later. The MM57109 can be halted when the RDY line is high by applying a logic "1" to the HOLD input before or at the rising edge of RDY. If HOLD goes high after RDY does, the processor will not stop until the next RDY pulse. Stopping the processor can allow more time for an external device to prepare data or an instruction. The MM57109 cannot be halted during execution of an instruction. For 2 word operations, the RDY line will go high twice, once for each fetch. The ISEL output is used during such operations to indicate when the processor is expecting an instruction; it will go low when data is expected. This is useful if the data and instructions are coming from two different sources.

DO_1 thru DO_4 are used to output data during an OUT operation. The number of digits and format depend on parameters set by software, especially the SMDC and TOGM instructions. The $R\overline{W}$ output is strobed low, once for each digit. Note in figure 3 that the MM57109 issues a second RDY pulse during the OUT operation. This is for external memory control and can be ignored if the processor is being used as a microcomputer peripheral.

The digit address lines, DA_1 thru DA_4, and the digit address strobe, DAS, are used to provide address information when the MM57109 is configured as a stand alone processor with its own memory. Typically, the digit address lines would provide the lower four bits of address with the upper four bits coming from an external read only memory.

Reference Source:

The source of the information used to design this circuit for the MM57109 is the National Semiconductor publication, MM57109 MOS/LSI Number Oriented Microprocessor, *copyright 1977 by National Semiconductor Corporation, published in March 1977. The publication number of this 24 page booklet is IM-B50M37. National Semiconductor Corporation is located at 2900 Semiconductor Dr, Santa Clara CA 95051, and the MM57109 is available from electronics distributors who handle National Semiconductor's product line.*

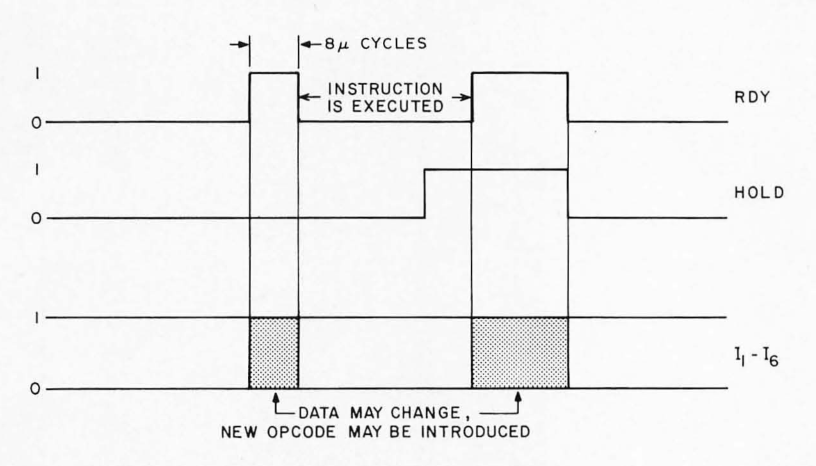


Figure 2: Instruction fetch and hold timing diagram. The RDY line goes high when the data is ready. If the HOLD line is also high RDY will remain high. When the HOLD line goes low, the RDY line will follow and the instruction will be executed. Adapted from figure 8c on page 10 of National Semiconductor Corporation's documentation, IM-B50M37 (March 1977).

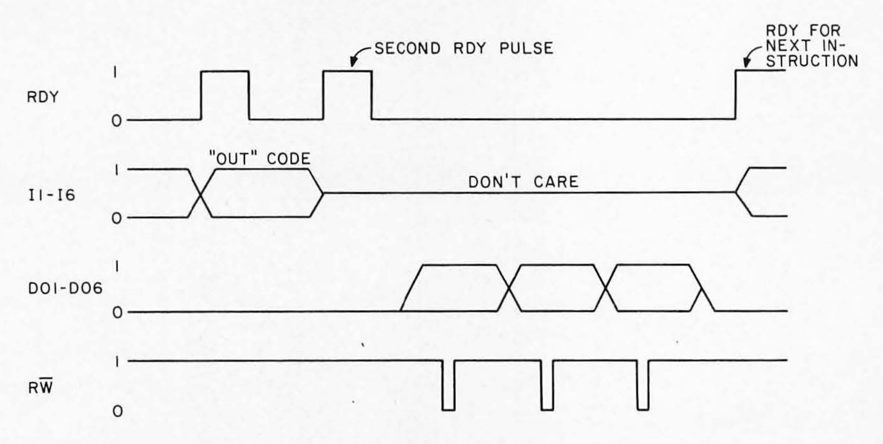


Figure 3: Timing diagram of an OUT instruction. The second RDY pulse can be ignored if the processor is being used as a peripheral to a computer.

The POR input is used to reset the processor after power is first applied. Following a 2 microcycle, or greater, positive pulse on this line, the MM57109 will issue three RDY signals. The first two should be ignored; processing begins following the third one.

The remaining outputs are all controlled by software. F_1 and F_2 may be set or pulsed by the SF or PF instructions. The ERROR line indicates an illegal operation or overflow and $\overline{BR}$ responds to a jump or branch operation with a pulse to "0."

Instruction Set

Table 1 details the MM57109's 70 instructions. This number is achieved with only a 6 bit word through the use of the INV instruction, octal 40, which gives double service to some of the other op codes. The instruction set provides a complete set of scientific calculator operations in

Table 1: The 70 command instruction set for the MM57109. The commands are broken into seven different classes: digit entry, data moves, math functions, clearing operations, branch functions, IO and mode control. Reproduced courtesy of National Semiconductor Corporation.

CLASS	SUBCLASS	MNEMONIC*	OCTAL OP CODE	FULL NAME	DESCRIPTION
Digit Entry		0	00	0	Mantissa or exponent digits. On first digit (d) the following occurs: Z → T Y → Z X → Y d → X See description of number entry on page 11.
		1	01	1	
		2	02	2	
		3	03	3	
		4	04	4	
		5	05	5	
		6	06	6	
		7	07	7	
		8	10	8	
		9	11	9	
		DP	12	Decimal Point	Digits that follow will be mantissa fraction.
		EE	13	Enter Exponent	Digits that follow will be exponent.
		CS	14	Change Sign	Change sign of exponent or mantissa. Xm = X mantissa Xe = X exponent CS causes −Xm → Xm or −Xe → Xe depending on whether or not an EE instruction was executed after last number entry initiation.
		PI	15	Constant π	3.1415927 → X, stack not pushed.
		EN	41	Enter	Terminates digit entry and pushes the stack. The argument entered will be in X and Y. Z → T Y → Z X → Y
		NOP	77	No Operation	Do nothing instruction that will terminate digit entry.
		HALT	17	Halt	External hardware detects HALT op code and generates HOLD = 1. Processor waits for HOLD = 0 before continuing. HALT acts as a NOP and may be inserted between digit entry instructions since it does not terminate digit entry.
Move		ROLL	43	Roll	Roll Stack. X T Y Z
		POP	56	Pop	Pop Stack. Y → X Z → Y T → Z 0 → T
		XEY	60	X exchange Y	Exchange X and Y. X ⟷ Y
		XEM	33	X exchange M	Exchange X with memory. X ⟷ M
		MS	34	Memory Store	Store X in Memory. X → M
		MR	35	Memory Recall	Recall Memory into X. M → X
		LSH	36	Left Shift Xm	X mantissa is left shifted while leaving decimal point in same position. Former most significant digit is saved in link digit. Least significant digit is zero.
		RSH	37	Right Shift Xm	X mantissa is right shifted while leaving decimal point in same position. Link digit, which is normally zero except after a left shift, is shifted into the most significant digit. Least significant digit is lost.

an easy to use keyboard entry format. The processor uses reverse Polish notation, RPN, which is the same system used on the Hewlett-Packard calculators. This method can obviate the need for parentheses in many cases.

There is a complete set of conditional branch test operations that may be quite useful if a programmable calculator or higher level language is being implemented. The BR output will pulse low if the condition being tested is true.

Among the numerous digit entry and IO operations are three different ways to input digits: AIN, IN and "digit as instruction" number entry, octal 00-11. This latter method is most similar to calculator number entry and is the input system assumed for the interface in this article.

The MM57109 offers two conventional

CLASS	SUBCLASS	MNEMONIC*	OCTAL OP CODE	FULL NAME	DESCRIPTION
Math	F (X,Y)	+	71	Plus	Add X to Y. X + Y → X. On +, −, x, / and YX instructions, stack is popped as follows: Z → Y T → Z 0 → T Former X, Y are lost.
		−	72	Minus	Subtract X from Y. Y − X → X
		x	73	Times	Multiply X times Y. Y x X → X
		/	74	Divide	Divide X into Y. Y ÷ X → X
		YX	70	Y to X	Raise Y to X power. $Y^X \rightarrow X$
	F (X,M)	INV +*	40, 71	Memory Plus	Add X to memory. M + X → M On INV +, −, x and / instructions, X, Y, Z, and T are unchanged.
		INV −*	40, 72	Memory Minus	Subtract X from memory. M − X → M
		INV x*	40, 73	Memory Times	Multiply X times memory. M x X → M
		INV /*	40, 74	Memory Divide	Divide X into memory. M ÷ X → M
	F (X) Math	1/X	67	One Divided by X	1 ÷ X → X. On all F (X) math instructions Y, Z, T and M are unchanged and previous X is lost.
		SQRT	64	Square Root	$\sqrt{X} \rightarrow X$
		SQ	63	Square	$X^2 \rightarrow X$
		10X	62	Ten to X	$10^X \rightarrow X$
		EX	61	E to X	$e^X \rightarrow X$
		LN	65	Natural log of X	ln X → X
		LOG	66	Base 10 log of X	log X → X
	F (X) Trig	SIN	44	Sine X	SIN(X) → X. On all F(X) trig functions, Y, Z, T, and M are unchanged and the previous X is lost.
		COS	45	Cosine X	COS(X) → X
		TAN	46	Tangent X	TAN(X) → X
		INV SIN*	40, 44	Inverse sine X	$SIN^{-1}(X) \rightarrow X$
		INV COS*	40, 45	Inverse cosine X	$COS^{-1}(X) \rightarrow X$
		INV TAN*	40, 46	Inverse tan X	$TAN^{-1}(X) \rightarrow X$
		DTR	55	Degrees to radians	Convert X from degrees to radians.
		RTD	54	Radians to degrees	Convert X from radians to degrees.
Clear		MCLR	57	Master Clear	Clear all internal registers and memory; initialize I/O control signals, MDC = 8, MODE = floating point. (See initialization.)
		ECLR	53	Error flag clear	0 → Error flag
Branch	Test	JMP*	25	Jump	Unconditional branch to address specified by second instruction word. On all branch instructions, second word contains branch address to be loaded into external PC.
		TJC*	20	Test jump condition	Branch to address specified by second instruction word if JC (I_6) is true (=1). Otherwise, skip over second word.
		TERR*	24	Test error	Branch to address specified by second instruction word if error flag is true (=1). Otherwise, skip over second word. May be used for detecting specific errors as opposed to using the automatic error recovery scheme dealt with in the section on Error Control.
		TX = 0*	21	Test X = 0	Branch to address specified by second instruction word if X = 0. Otherwise, skip over second word.
		TXF*	23	Test |X| < 1	Branch to address specified by second instruction word if |X| < 1. Otherwise, skip over second word. (i.e. branch if X is a fraction.)
		TXLT0*	22	Test X < 0	Branch to address specified by second instruction word if X < 0. Otherwise, skip over second word.

microprocessor operations, HALT and NOP, but with a difference. HALT, by itself, only acts as a NOP; it does not stop the machine. It is designed to be detected with external hardware that will generate a HOLD signal to halt the processor.

The number of microcycles required to execute an instruction may vary from a few hundred to many thousands as shown in table 2. Speaking generally, the complex operations such as trigonometric and logarithmic functions take the greatest time. As a bench mark, the floating point addition time is 2200 microcycles, or 22 ms. However the floating point add time might not make a very reliable benchmark since it varies over a wide range (22 ms, typical, to 66 ms, worst case) depending on the numbers involved.

Figure 4 shows how easy it is to interface the MM57109 to your system. Most of the required pins are TTL compatible. The POR and HOLD inputs, however, must have a

Table 1, continued:

CLASS	SUBCLASS	MNEMONIC*	OCTAL OP CODE	FULL NAME	DESCRIPTION
Branch	Count	IBNZ	31	Increment memory and branch if M ≠ 0	M + 1 → M. If M = 0, skip second instruction word. Otherwise, branch to address specified by second instruction word.
		DBNZ	32	Decrement memory and branch if M ≠ 0	M − 1 → M. If M = 0, skip second instruction word. Otherwise, branch to address specified by second instruction word.
I/O	Multi-digit	IN*	27	Multidigit input to X	The processor supplies a 4-bit digit address (DA4—DA1) accompanied by a digit address strobe ($\overline{DAS}$) for each digit to be input. The high order address for the number to be input would typically come from the second instruction word. The digit is input on D4—D1, using ISEL = 0 to select digit data instead of instructions. The number of digits to be input depends on the calculation mode (scientific notation or floating point) and the mantissa digit count (See Data Formats and Instruction Timing). Data to be input is stored in X and the stack is pushed (X → Y → Z → T). At the conclusion of the input, DA4—DA1 = 0.
		OUT*	26	Multidigit output from X	Addressing and number of digits is identical to IN instruction. Each time a new digit address is supplied, the processor places the digit to be output on DO4—DO1 and pulses the R/$\overline{W}$ line active low. At the conclusion of output, DO4—DO1 = 0 and DA4—DA1 = 0.
I/O	Single-digit	AIN	16	Asynchronous Input	A single digit is read into the processor on D4—D1. ISEL = 0 is used by external hardware to select the digit instead of instruction. It will not read the digit until $\overline{ADR}$ = 0 (ISEL = 0 selects $\overline{ADR}$ instead of I_5), indicating data valid. F2 is pulsed active low to acknowledge data just read.
I/O	Flags	SF1	47	Set Flag 1	Set F1 high, i.e. F1 = 1.
		PF1	50	Pulse Flag 1	F1 is pulsed active high. If F1 is already high, this results in it being set low.
		SF2	51	Set Flag 2	Set F2 high, i.e. F2 = 1.
		PF2	52	Pulse Flag 2	F2 is pulsed active high. If F2 is already high, this results in it being set low.
		PRW1	75	Pulse R/$\overline{W}$ 1	Generates R/$\overline{W}$ active low pulse which may be used as a strobe or to clock extra instruction bits into a flip-flop or register.
		PRW2	76	Pulse R/$\overline{W}$ 2	Identical to PRW1 instruction. Advantage may be taken of the fact that the last 2 bits of the PRW1 op code are 10 and the last 2 bits of the PRW2 op code are 01. Either of these bits can be clocked into a flip-flop using the R/$\overline{W}$ pulse.
Mode Control		TOGM	42	Toggle Mode	Change mode from floating point to scientific notation or vice-versa, depending on present mode. The mode affects only the IN and OUT instructions. Internal calculations are always in 8-digit scientific notation.
		SMDC*	30	Set Mantissa Digit Count	Mantissa digit count is set to the contents of the second instruction word (=1 to 8).
		INV	40	Inverse Mode	Set inverse mode for trig or memory function instruction that will immediately follow. Inverse mode is for next instruction only.



The LATEST in Tape Systems



MODEL CC-8
$185.00 (4800 Baud)
$195.00 (9600 Baud and 220V/50 Hz)

9600 BAUD CASSETTE RECORDER

An ASYNCHRONOUS NRZ type Recorder with remote motor start/stop. Error rate 10^8 at 4800 BAUD. Can be used from 110 to 9600 BAUD into a UART — no clocking required. This is not an audio recorder. It takes RS232 or TTL signals from the terminal or computer and gives back the same signals. No audio interface is used. Motor start/stop is manual or through TTL or RS232 signals.

Tape speeds are 1.6" / 3.0" and 6.0" per second. 110 volt, 60 Hz, 5 watts. (220 Volts on special order). Can use high quality audio cassettes (Philips Type) or certified data cassettes. Can be used in remote locations from a 12 Volt battery.

Recommended for DATA LOGGING, WORD PROCESSING, COMPUTER PROGRAM RELOADING and DATA STORAGE. Manual control except for motor start/stop. 6800, 8080 or Z80 software for file or record searching available on request with order. Used by major computer manufacturers, Bell Telephone and U.S. Government for program reloading and field servicing.

AVAILABILITY — Off the shelf.

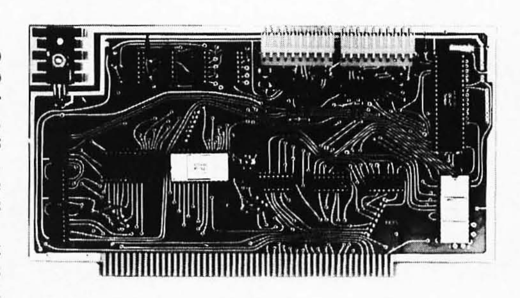
PROVIDES MONITOR AND TAPE SOFTWARE in ROM. TERMINAL and TAPE PORTS on SAME BOARD. CONTROLS ONE or TWO TAPE UNITS (CC-8 or 3M3B).

This is a complete 8080, 8085, or Z80 system controller. It provides the terminal I/O (RS232, 20 mA or TTL) and the data cartridge I/O, plus the motor controlling parallel I/O latches. Two kilobytes of on board ROM provide turn on and go control of your Altair or IMSAI. NO MORE BOOTSTRAPPING. Loads and Dumps memory in hex on the terminal, formats tape cartridge files, has word processing and paper tape routines. Best of all, it has the search routines to locate files and records by means of six, five, and four letter strings. Just type in the file name and the recorder and software do the rest. Can be used in the BiSync (IBM), BiPhase (Phase encoded) or NRZ modes with suitable recorders, interfaces and software.

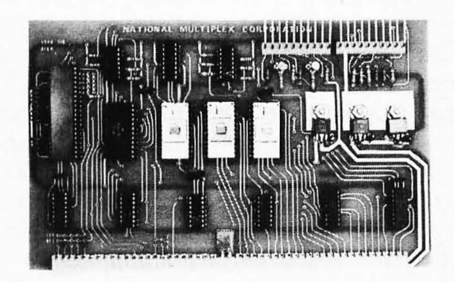
This is Revision 8 of this controller. This version features 2708 type EPROM's so that you can write your own software or relocate it as desired. One 2708 preprogrammed is supplied with the board. A socket is available for the second ROM allowing up to a full 2K of monitor programs.

Fits all S100 bus computers using 8080 or Z80 MPU's. Requires 2 MHz clock from bus. Cannot be used with audio cassettes without an interface. Cassette or cartridge inputs are TTL or RS232 level.

AVAILABILITY — Off the shelf.



2SIO (R) CONTROLLER
$190.00, Tested & Assmb.



6800 CONTROLLER for SWTP
$190.00, Tested & Assmb.

PROVIDES MONITOR AND TAPE SOFTWARE in EPROM. EXPANDS MIKBUG with 1K of ADDITIONAL ROM PROGRAM

This is a complete tape controller for the SWTP 6800 system. Has 3K of EPROM space for your own programs. A 1 K ROM (2708) is provided with all tape and monitor functions. The ROM program is identical to our extensive 8080 ROM program.

Has one ACIA for one or two tape drives, one UART for an additional Serial port and a 4 bit parallel port for motor control. Will control one or two CC-8 or 3M3B drives with the software provided. Can be used with other tape drives controllable with 4 TTL bits if appropriate software changes are made.

Extra serial port is provided for your use with a second terminal or printer (RS232, TTL or 20 ma).

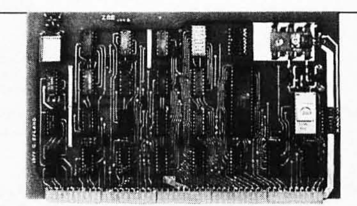
The ROM program supplements the MIKbug program and is entered automatically on reset. SWTbug compatible ROM is also available.

AVAILABILITY — Off the shelf.

Z 80 BOARD for SWTP COMPUTER

Now you can use the 8080/Z80 software programs in your SWTP 6800 machine. Replaces your MPU board with a Z80 and ROM so that you are up and running with your present SWTP memory and MPS card. 1 K ROM on board replaces MIKBUG.

AVAILABILITY — Off the shelf.



$190.00, Tested & Assmb.

For U.P.S. delivery, add $3.00. Overseas and air shipments charges collect, N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and BankAmericard accepted.

National Multiplex Corporation

3474 Rand Avenue, South Plainfield NJ 07080 Box 288 Phone (201) 561-3600 TWX 710-997-9530

INSTRUCTION MNEMONIC	EXECUTION TIME (MICROCYCLES) (AVERAGE)	EXECUTION TIME (MICROCYCLES) (WORST-CASE VALUES)	INSTRUCTION MNEMONIC	EXECUTION TIME (MICROCYCLES) (AVERAGE)	EXECUTION TIME (MICROCYCLES) (WORST-CASE VALUES)
0–9		238	OUT		583
DP		152	IN		395
EE		151	SF1		163
CS		166	PF1		185
PI		1312	SF2		163
HALT		134	PF2		185
AIN		284	PRW1		130
TJC		208	PRW2		130
TX=0		278	SIN	56200	95900
TXLT0		197	COS	56200	95900
TXF		277	TAN	35000	97600
TERR		191	INV SIN	54000	93900
JMP		186	INV COS	54000	93900
IBNZ		2314	INV TAN	30200	92900
DBNZ		2314	LN	24800	92000
SMDC		163	LOG	30700	92600
XEM		812	EX	30800	93900
MS		839	10X	27400	96500
MR		1385	+, –	2200	6600
LSH		168	INV+, INV–	1700	5000
RSH		173	(M+, M–)		
INV		166	x	3200	22700
EN		552	INV x (MX)	2700	21400
TOGM		157	/	7800	22300
ROLL		905	INV / (M/)	7300	21100
ECLR		163	1/X	4500	22800
POP		448	YX	55400	95500
MCLR		734	SQRT	7000	30200
XEY		652	SQ	3000	21900
NOP		122	DTR, RTD	9600	41700

Table 2: Execution times for command set. The execution time is measured in microcycles which are defined as being 10 µs long. Reproduced courtesy of National Semiconductor Corporation.

IN/OUT Instructions (a) Mode = Scientific Notation

DA4–DA1	IN: / OUT:	D4 / DO4	D3 / DO3	D2 / DO2	D1 / DO1
0		Most significant exponent digit			
1		Least significant exponent digit			
2		Sm	0	0	Se
3		Not used			
4		Most significant mantissa digit (Decimal point follows this digit)			
.		.			
.		.			
.		.			
MDC + 3		Least significant mantissa digit			

IN/OUT Instructions (b) Mode = Floating Point

DA4–DA1	DP POS	IN: / OUT:	D4 / DO4	D3 / DO3	D2 / DO2	D1 / DO1
2			Sm	0	0	0
3			DP POS			
4	11		Most significant Mantissa Digit = 0–9			
5	10		.			
.	.		.			
.	.		.			
.	.		.			
MDC + 3	12 – MDC		Least significant Mantissa Digit = 0–9			

Table 3: The data format for floating point and scientific notation input and output. MDC stands for mantissa digit count which is set by the SMDC instruction. It is initially set to 8. Sm is the sign of the mantissa; it is 0 for positive and 1 for negative numbers. Se is the sign of the exponent which is set to 0, for positive, in the floating point mode. DP POS is the decimal point position indicator which is a value in the range from 11 to 12-MDC, which indicates a digit, as given by the DP POS column in the table. The decimal point is located to the right of this digit.

voltage swing from –4 V to +5 V. Fortunately, this is easily achieved with an LM339 comparator which makes a fine TTL to MOS level shifter and is widely available. The $\overline{BR}$ and $R\overline{W}$ lines are the only required outputs that cannot drive TTL directly. CMOS inverters and flip flops are used to latch pulses from these lines anyway, but in this case they also provide buffering.

The 2.2 k pull down resistors to V_{DD} from the ERROR, RDY and DO lines assume the use of a bipolar (TTL) input port. If your input port lines use high impedance MOS receivers, then these resistors should be replaced by 15 k resistors to ground, 0V.

It is worth mentioning that in this circuit there are a number of outputs of the MM57109 that are completely unused. This is because the MM57109 is a very versatile device that can be configured in several different ways. In this application the computer performs many of the functions of the unused pins.

The simple program detailed in listing 1 is designed to allow you to become familiar with the MM57109 by giving it one instruction at a time and single stepping its operation. A number of useful subroutines are included that may be applied to larger applications programs. After each colon prompt character is displayed you are expected to supply a two digit octal number corresponding to the instruction you wish to execute. If it is a 2 word instruction, the second word is entered after the next colon. Whenever the OUT instruction, octal 26, is used, the computer goes to a routine that reads the data into a buffer and then dumps it to the display "as is." This way, the effect that different instructions have on the display format shown in table 3 can be seen. Before requesting an instruction, the computer polls the ERROR and $\overline{BR}$ lines and outputs an "E" or "B," respectively, if these lines are active. Once the ERROR line is set it must be cleared with the ECLR instruction, but the $\overline{BR}$ line is reset automatically by the program.

When the program is first entered at 0200 it initializes the processor by outputting a POR pulse and ignoring the first two RDY pulses before halting on the third. Halting for instruction input is done by bringing the HOLD line high. When an instruction is first moved to the output port, the HOLD line is left high. It is brought down on a separate command so that the data on I_1 thru I_6 is fully stable before HOLD starts to change. The computer senses when it is time for a new instruction by monitoring the RDY line

the electric pencil II™

© 1978 Michael Shrayer

The Electric Pencil II is a Character Oriented Word Processing System. This means that text is entered as a string of continuous characters and is manipulated as such. This allows the user enormous freedom and ease in the movement and handling of text. Since line endings are never delineated, any number of characters, words, lines or paragraphs may be inserted or deleted anywhere in the text. The entirety of the text shifts and opens up or closes as needed in full view of the user. The typing of carriage returns as well as word hyphenation is not required since lines of text are formatted automatically.

As text is typed in and the end of a screen line is reached, a partially completed word is shifted to the beginning of the following line. Whenever text is inserted or deleted, existing text is pushed down or pulled up in a wrap around fashion. Everything appears on the video display screen as it occurs which eliminates any guesswork. Text may be reviewed at will by variable speed scrolling both in the forward and reverse directions. By using the search or the search and replace function, any string of characters may be located and/or replaced with any other string of characters as desired.

When text is printed, **The Electric Pencil II** automatically inserts carriage returns where they are needed. Numerous combinations of line length, page length, line spacing and page spacing allow for any form to be handled. Character spacing, **BOLD FACE**, multicolumn as well as bidirectional printing are included in the Diablo versions. Right justification gives right-hand margins that are even. Pages may be numbered as well as titled. This entire page (excepting the large titles and logo) was printed by the Diablo version of **The Electric Pencil II** in one pass.

Now on CP/M

You've probably seen **The Electric Pencil** in action by now. It's the most powerful 8080/Z80 character oriented word processor on the market today. Michael Shrayer is now proud to present the new **Electric Pencil II**.

NEW FEATURES: !!! CP/M Compatible !!! Disk Operating System Supports Four Disk Drives !!! Simple File Management !!! Quick and Easy Disk Storage and Retrieval !!! Dynamic Print Formatting !!! Multicolumn Printing !!! Print Value Chaining !!! Page-at-a-time Scrolling !!! New Bidirectional Multispeed Scrolling Controls !!! New Subsystem with Print Value Scoreboard !!! Automatic Word and Record Number Tally !!! Cassette Backup Capability !!! Full Margin Control !!! End-of-Page Control !!! Non-Printing Text Commenting !!! Line and Paragraph Indentation !!! Centering !!! Underlining !!! **BOLD FACE** !!!

W I D E S C R E E N V I D E O ! ! !
Available to Imsai VIO video users for a huge 80x24 character screen !!

HAVE WE GOT A VERSION FOR YOU ?

The Electric Pencil II operates with any 8080/Z80 based microcomputer that supports a CP/M disk system and uses a Imsai VIO, Processor Technology VDM-1, Polymorphic VTI, Solid State Music VB-1B, Vector Graphic Flashwriter or any similar memory mapped video interface. Specify when using CP/M that has been modified for Micropolis or North Star disk systems as follows: For North Star add suffix A to version number, for Micropolis add suffix B to version number, e.g. SS-IIA, DV-IIB.

Vers.	Video	Printer	Price
SS-II	SOL	TTY or similar	$225.
SP-II	VTI	TTY or similar	$225.
SV-II	VDM	TTY or similar	$225.
SI-II	VIO	TTY or similar	$250.
DS-II	SOL	Diablo 1610/20	$275.
DP-II	VTI	Diablo 1610/20	$275.
DV-II	VDM	Diablo 1610/20	$275.
DI-II	VIO	Diablo 1610/20	$300.



MICHAEL SHRAYER SOFTWARE
1253 Vista Superba Drive
Glendale, CA 91205
(213) 956-1593

The Electric Pencil I is still available for non CP/M users:

Vers.	Video	Printer	Cassette	Disk Drive	Price
SS	SOL	TTY or similar	CUTS	---	$100.
SP	VTI	TTY or similar	Tarbell	---	$100.
SV	VDM	TTY or similar	Tarbell	---	$100.
SSN	SOL	TTY or similar	CUTS	North Star	$125.
SPN	VTI	TTY or similar	Tarbell	North Star	$125.
SVN	VDM	TTY or similar	Tarbell	North Star	$125.
DS	SOL	Diablo 1610/20	CUTS	---	$150.
DP	VTI	Diablo 1610/20	Tarbell	---	$150.
DV	VDM	Diablo 1610/20	Tarbell	---	$150.
DSN	SOL	Diablo 1610/20	CUTS	North Star	$175.
DPN	VTI	Diablo 1610/20	Tarbell	North Star	$175.
DVN	VDM	Diablo 1610/20	Tarbell	North Star	$175.

coming attractions

Sort & Merge Utility !!!

The NEC printer package !!!

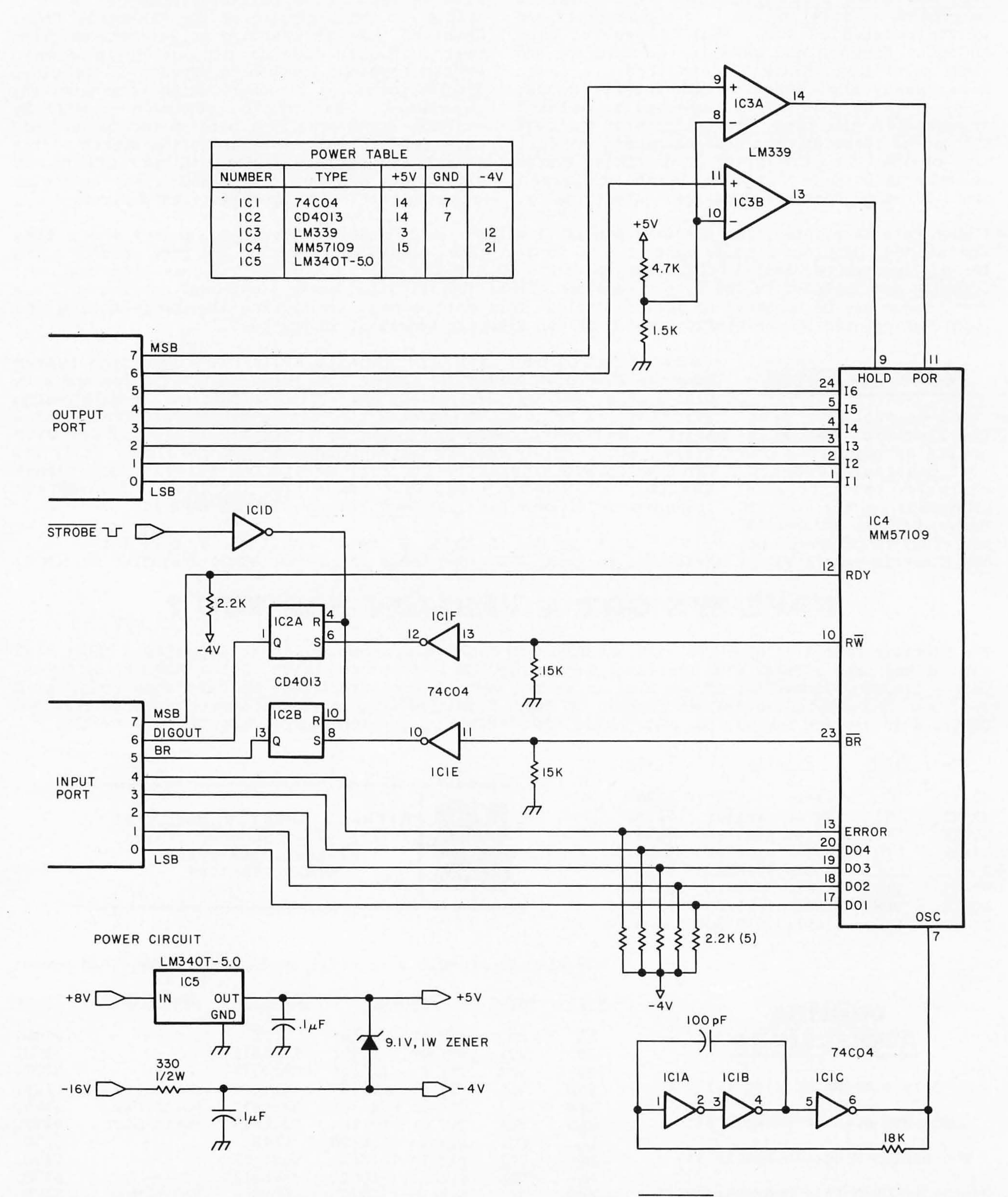
The HELIOS Electric Pencil !!!

Pencil to CP/M file conversion !!!

CP/M to Pencil file conversion !!!

Demand a demo from your dealer!

Circle 316 on inquiry card.



POWER TABLE				
NUMBER	TYPE	+5V	GND	-4V
IC1	74C04	14	7	
IC2	CD4013	14	7	
IC3	LM339	3		12
IC4	MM57109	15		21
IC5	LM340T-5.0			

Figure 4: Schematic diagram for interfacing the processor to a computer. The $\overline{STROBE}$ signal can be any software controlled output pulse with a width varying from 200 ns to 50 μs. The rise and fall times are noncritical. All unused inputs in the circuit should be grounded to prevent floating voltage problems. All resistors are 0.25 W unless otherwise specified. All resistances are measured in ohms.

Listing 1: An 8080 familiarization program for the MM57109 which allows the user to execute single commands and observe the results. The program will output a prompting colon after which the operator can input the command to be observed in its octal format.

```
;SOFTWARE LISTING FOR MM57109 MICROPROCESSOR DRIVER/DEMONSTRATOR
;WRITTEN FOR AN 8080  BY P NELSON
0200 3E BF                MVI A,BF     ;SET POR=1,HOLD=0
0202 32 01 80             STA I/O#1    ;INSTRUCTION = NOP
0205 0E 04                MVI C,04     ;SHORT TIMING VALUE
0207 0D          LOOP     DCRC         ;TO TIME POR
0208 C2 07 02             JNZ LOOP     ;PULSE
020B 79                   MOV A,C      ;CLR A
020C 32 01 80             STA I/O#1    ;SET HOLD, POR=0
020F CD 8F 02             CALL NXTRDY ;WAIT FOR RDY PULSE
0212 CD 8F 02             CALL NXTRDY ;WAIT FOR RDY PULSE
0215 F6 40       NEXT     ORI 40       ;SET HOLD=1 SO 57109
0217 32 01 80             STA I/O#1    ;STOPS ON NEXT RDY
021A CD 8F 02             CALL NXTRDY ;RETURNS WITH 57109 HALTED
021D 3A 01 80 NEXT+       LDA I/O#1    ;READ BR,ERROR
0220 17                   RAL
0221 17                   RAL          ;BR TO CARRY
0222 17                   RAL
0223 32 04 80             STA I/O#4    ;CLR F.F.'S WITH STROBE
0226 47                   MOV B,A      ;TEMP STORE ACC
0227 D2 2F 02             JNC ERROR?   ;IF NO BRANCH
022A 3E 42                MVI A,42     ;ASCII B
022C CD 10 01             CALL OUTCHR ;CHARACTER DISPLAY
022F 78          ERROR?   MOV A,B      ;RETRIEVE ACC
0230 17                   RAL          ;ERROR TO CARRY
0231 D2 39 02             JNC FETCH    ;IN NO ERROR
0234 3E 45                MVI A,45     ;ASCII E
0236 CD 10 01             CALL OUTCHR ;CHARACTER DISPLAY
0239 3E 3A       FETCH    MVI A,3A     ;ASCII :
023B CD 10 01             CALL OUTCHR ;CHARACTER DISPLAY
023E CD 9E 02             CALL OCTIN   ;GET INSTRUCTION CODE
0241 CD AE 02             CALL INSTRCT;GIVE 57109 OPCODE
0244 FE 16                CPI 16       ;OUT CODE
0246 CA 4C 02             JZ OUT       ;WAS IT AN OUT OP
0249 C3 15 02             JMP NEXT     ;IF NOT, NEXT INSTRUCTION


; OUT READS IN CORRECT # OF DIGITS, STOPPING BY SENSING RDY AND
;THEN DUMPS BUFFER TO DISPLAY IN HEX/BCD
024C 21 0C 03 OUT         LXI H        ;TOP OF BUFFER LOCATION
024F 3A 01 80 NOTYET      LDA I/O#1    ;GET INPUT
0252 17                   RAL          ;ROTATE
0253 17                   RAL          ;DIGOUT INTO CARRY
0254 02 4F 02             JNC NOTYET   ;DIGOUT=0?
0257 1F          NXTDIG   RAR          ;RESTORE DIGIT
0258 1F                   RAR          ;LOCATION IN ACC.
0259 77                   MOV M,A      ;PUT IN BUFFER
025A 20                   DCR L        ;DCR POINTER
025B 32 04 80             STA I/O#4    ;RESET DIGOUT F.F.
025E 3E 40                MVI A,40     ;NOW THAT WE ARE OAST OUT'S 2ND
0260 32 01                STA I/O#1    ;RDY,SET HOLD=1 TO STOP AT END OF OUT
0263 3A 01 80 AGAIN       LDA I/O#1    ;GET INPUT
0266 17                   RAL          ;ROTATE RDY INTO CARRY
0267 DA 71 02             JC DISPLAY   ;IF DONE, GO TO DISPLAY
026A 17                   RAL          ;ROTATE DIGOUT INTO CARRY
026B D2 63 02             JNC AGAIN    ;NO DIGIT READY?
026E C3 57 02             JMP          ;STORE DIGIT,GET NEXT ONE
0271 3E 0C       DISPLAY MVI A,0C      ;CALCULATE DIGIT
0273 95                   SUBL         ;COUNT(0C POINTER VALUE)
0274 4F          LISTING MOV C,A       ;STORE COUNT IN REGISTER C
0275 21 0C 03             LXI H        ;BUFFER LOCATION
0278 7E                   MOV A,M      ;GET 1ST DIGIT
0279 E6 0F                ANI 0F       ;GET LOWER 4 BITS
027B FE 0A                CPI 0A
027D DA 82 02             JC "0-9"     ;DON,T CHANGE IF 0-9
0280 C6 07                ADI 07       ;CONVERT A-F
0282 C6 30       "0-9"    ADI 30       ;CONVERT ALL TO ASCII
0284 CD 10 01             CALL OUTCHR ;DISPLAY CHARACTER
0287 2D                   DCR L        ;DECR POINTER
0288 0D                   DCR C        ;DECR COUNT
0289 C2 78 02             JNZ MORE     ;DMORE DIGITS TO DISPLAY
028C C3 1D 02             JMP NEXT+    ;DO NEXT INSTRUCTION


;NXTRDY WAITS FOR A POSITIVE TRANSITION ON THE RDY LINE AND THEN
;RETURNS
028F 3A 01 80 NXTRDY      LDA I/O#1    ;GET INPUT
0292 17                   RAL          ;ROTATE RDY INTO CARRY
0293 DA 8F 02             JC NXTRDY
0296 3A 01 80 NOTRDY      LDA I/O#1    ;GET INPUT
0299 17                   RAL          ;ROTATE RDY INTO CARRY
029A D8                   RC           ;IF CARRY IS HIGH,RETURN
029B C3 96 02             JMP NOTRDY   ;OTHERWISE TRY AGAIN


;OCTIN RECEIVES 2 DIGITS (IT ASSUMES NUMERIC INPUT) AND CONVERTS
;TO A 6 BIT VALUE IN ACC REPRESENTING A 2 OCTAL DIGIT INSTRUCTION
;CODE FOR THE 57109.  AN INTERRUPT DRIVEN KEYBOARD IS ASSUMED,BUT
;THE 3 BYTE FORMAT (EI,HLT,NOP) LEAVES ROOM FOR A CALL INSTRUCTION
;TO THE USER'S INPUT SUBROUTINE IF THIS IS NOT THE CASE
029E FB                   EI           ;GET
029F 76                   HLT          ;BYTE
020A 00                   NOP          ;INPUT
02A1 E6 07                ANI 07       ;MASK FOR LOWER 3 BITS
02A3 17                   RAL          ;MOVE
02A4 17                   RAL          ;DIGIT UP 3
02A5 17                   RAL          ;POSITIONS
02A6 47                   MOV B,A      ;TEMP STORE ACC
02A7 FB                   EI           ;GET
02A8 76                   HLT          ;BYTE
02A9 00                   NOP          ;INPUT
02AA E6 07                ANI 07       ;MASK FOR LOWER 3 BITS
02AC B0                   ORA B        ;COMBINE DIGITS
02AD C9                   RET          ;(RETURN)


;INSTRCT PUTS INSTRUCTION CODE ON I1 TO I6 WHILE LEAVING HOLD HIGH.
;IT THEN SETS HOLD TO 0.
02AE F6 40       INSTRCT ORI 40        ;HOLD=1 WITH OPCODE
02B0 32 01 80             STA I/O#1    ;OUTPUT IT
02B3 E6 3F                ANI 3F       ;MXB-1 OF HOLD = 0
02B5 32 01 80             STA I/O#1    ;OUTPUT IT
02B8 C9                   RET          ;(RETURN)
```

with the input port. If you have a spare interrupt, however, you can connect it to the RDY line to free the computer from the task of polling the input port while the MM57109 is executing an instruction.

DIGOUT is monitored by the computer when processing the MM57109's OUT instruction. It goes high to signify the presence of a digit on DO_1 thru DO_4 and must be reset by the computer after the digit is read in. When reading in digits, the computer will also monitor the RDY line to tell when the instruction is completed.

It should be noted that this program was written for my personal 8080 system and uses memory mapped IO, ie: my IO ports are addressed as memory locations. If you have a system which uses the 8080 IO ports, you will want to substitute IN and OUT instructions for LDA and STA, respectively.■

A COMPLETELY REFURBISHED "SELECTRIC" ASCII TERMINAL FOR THE SMALL BUSINESSMAN OR SERIOUS HOBBYIST.

The AJ 841 I/O Terminal. Now with RS 232 interface. Now available from dealers.

Demand for our AJ 841 I/O computer terminal has been great. And now that we've finally added RS 232 interface, it will get even greater. So while you can still order it direct from us, now you can also buy it from dealers around the country. But act now. Our special introductory price of $995 will soon be raised.

The AJ 841 features:

- Choice of Serial RS 232 or parallel interface
- ASCII code
- 14.9 cps printout
- High quality Selectric printing
- Reliable, heavyduty Selectric mechanism
- Off-line use as typewriter
- Documentation included
- 30-day warranty — parts and labor
- Mail order shipments to AJ office *or* direct to you.

Warranty and service available from your local AJ dealer — or from AJ offices in:

Atlanta/Boston/Chicago/Cincinnati/ Cleveland/Columbus/Dallas/ Detroit/Hackensack/Houston/ Los Angeles/New York/Philadelphia/ San Jose/Washington, D.C.

Full warranty information available on request.

Order with coupon—or from your local AJ dealer

For location of your nearest AJ dealer call *toll-free:* 800/538-9721 (Calif. residents call 408/263-8520).



HOW TO ORDER DIRECT FROM THE FACTORY

1. Make cashier's check or money order payable to Anderson Jacobson, Inc. and mail to:
   Anderson Jacobson, Inc.
   PERSONAL COMPUTER TERMINAL
   521 Charcot Avenue
   San Jose, CA 95131
2. You will be notified when your unit is ready for shipment. Allow six to eight weeks for delivery.
3. Shipments direct to you will involve a collect freight charge (FOB San Jose) in addition to the $35 handling charge. Shipments to nearest AJ office include freight in the $35 handling charge.
4. For warranty or repair service, return unit to local dealer or to designated AJ service location.

Available only in U.S.

CLIP AND MAIL WITH ORDER

Select interface: ☐ RS 232 or ☐ Parallel
Select keyboard: ☐ EBCD or ☐ Correspondence

☐ Ship direct to me (freight collect, FOB San Jose)
☐ Ship to AJ office (freight incl. in handling charge)

Number of units ________ at $995 each $ ________
Sales tax at delivery location $ ________
Shipping and handling $35 each (excluding San Jose) $ ________
TOTAL $ ________

NAME ________
ADDRESS ________
CITY ________ STATE ________ ZIP ________
PHONE ( ) ________

Circle 10 on inquiry card.

"THE ORIGINAL" Personal Computing 78™

PERSONAL COMPUTING COLLEGE
Has the industry's leading speakers participating

Dr. John W. Mauchly, Co-inventor of ENIAC.
Robert W. Bemer, The father of ASCII
Dr. Adam Osborne, Author, of Osborne & Associates
Robert S. Jones, Publisher & Editor-in-chief, INTERFACE AGE MAGAZINE
Carl Warren, Senior Editor, INTERFACE AGE MAGAZINE
Walter Banks, Author, University of Waterloo
Dr. Christopher A. Titus, Author of the BUGBOOKS
Carl T. Helmers, Jr., Editor-in-chief, BYTE MAGAZINE
Christopher P. Morgan, Senior Editor, BYTE MAGAZINE
Blaise W. Liffick, Senior Editor, BYTE MAGAZINE
Dr. Robert Suding, of The Digital Group
Chod Harris, of the Americal Radio Relay League
David H. Ahl, Publisher, CREATIVE COMPUTING MAGAZINE
Merl Miller, Author, Publisher dilithum Press/Matrix
Bill Langenes, Associate Editor, COMPUTER RETAILING
Jim Warren, West Coast Computer Faire
Sol Libes, President, Amateur Computer Group of New Jersey
Richard Moberg, President, Philadelphia Area Computer Society
Richard A. Kuzmack, President, Chesapeake Microcomputer Club
Rodnay Zaks, Author, Publisher, SYBEX Inc.
Howard A. Chamberlin, Jr. (Hal), Author, Inventor, Micro Technology Unlimited
Larry Steckler, Editor, Radio-Electronics Magazine
Tod Loofbourrow, Student and Author for Interface Age Magazine and Hayden Book Co.
many-many more.

80 hours of free seminars on subjects such as:

- Building your own robot
- Personal computing applications for the home
- The microcomputer controlled solar energy home
- Business applications
- Educational uses
- Understanding softwear
- Ham radio applications
- Designing your own microcomputer system
- Everything you wanted to know about floppies
- Softwear copywriting and trademarks

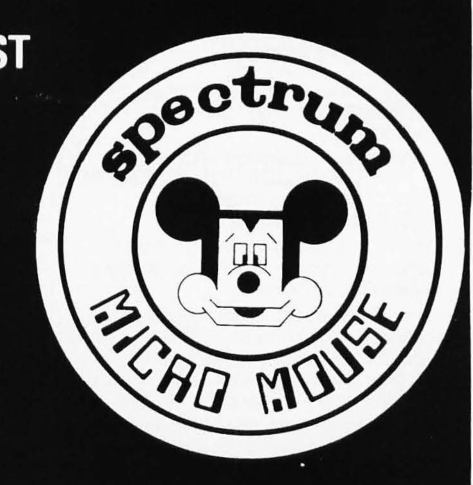
SPECIAL at our show:

- The unveiling of a new, complete, easy-to-learn high level language for micro-computers.
- A complete series of seminars aimed at the small business man presented by the INTERFACE AGE speaking team.
- Computer music.

THE AMAZING MICRO-MOUSE MAZE CONTEST

Official East Coast Runoffs being held Fri., Sat., Sun. at the show. Sponsored by IEEE Spectrum.

A contest of self-contained microcomputer controlled mice designed to negotiate a complex maze.



PHILADELPHIA CIVIC CENTER

August 25, 26, 27th at the Philadelphia Civic Center

Circle 302 on inquiry card.

See our ad on page 151.

PASCAL

A Structurally Strong Language

Stephen R Alpert
Worcester Polytechnic Institute
Worcester MA 01609

PASCAL forces the user to think logically and plan out the program.

Most PASCAL users consider programs better if they have fewer labels.

People should be able to communicate their ideas to a computer in a language that people understand; not simply in a language they know. Additionally, if the computer can be made to understand the same language easily, all the more reason to consider its use. Such a language is PASCAL. This language, perhaps more than any other common language, is the easiest to understand and more importantly, allows a straightforward presentation of most algorithms. Although many languages also make this claim, few have the overwhelming and energetic support from collegiate computer science departments. Let's consider some of the language features of PASCAL.

This language is equipped with a precise syntactical description that defines both how programs may be constructed and how PASCAL compilers should function. There is a required form for programs, statements within programs, and data operated upon by programs. At first glance, a naive user may rebel at this apparent lack of freedom: (eg: BASIC allows a dimension statement virtually anywhere in a program). One soon learns that this structure admits very general programs and in no way limits the programmer in exercising his talents. On the contrary, it forces the user to think logically and plan out the program.

A program written in PASCAL may utilize the free format form of programs that is conducive to structured programming. Unlike line oriented source languages, PASCAL allows extra spaces, tabs and carriage controls to be inserted anywhere without significance except in the middle of identifiers or character strings. Comments may be inserted wherever spaces may be inserted and are delimited by "(* ... *)". A program is made up of two parts, a heading and a block. The heading contains the name of the program and lists its parameters. The parameters are somewhat implementation dependent but normally specify the names of file pointers from which the default input is received and to which output is sent. A typical heading is

```
program parser (input, output)
```

A block consists of six separate segments or sections of a program. All but the last part are optional. These are:

- Label declaration section
- Constant declaration section
- Type declaration section
- Variable declaration section
- Procedure and function declaration section
- Statement section

Labels in PASCAL identify statements to which control may be transferred. Labels are numeric; more specifically, unsigned integers. Not every statement requires a label. In fact, most PASCAL users consider programs better if they have fewer labels. At first glance, these declarations might seem a nuisance, but they force the user to think about the entire program before sitting down at a terminal.

The constant declarations allow a user to create synonyms for constants used in the program. Thus

```
const   pi=3.141592;
        e=2.7182818;
```

defines the constants "pi" and "e" for use throughout the program. Clearly, it no longer is necessary to type 3.141592 in the several places required by a program. Additionally, one may name character strings as well

```
const title='matrix inversion program v01';
```

The type declaration section allows creation of user defined named data types. This will be discussed in some detail later. PASCAL has four predefined data types: integer, real, Boolean, and character. Most versions of BASIC support the first three types and

ARTEC CRAFTSMANSHIP HAS CREATED

The First Truly Silent Motherboard

Noise in your bus lines means errors in your programs. The Artec shielded Motherboard totally eliminates noise.

At 4MHz, the Artec shielded Motherboard is free from spurious noise. No ringing in your bus lines. No errors in your programs.

This Motherboard offers you engineering and craftsmanship never before available in the small computer field. Outstanding as either a replacement for your present Motherboard or as the heart of a new system. Consider these features:

- ⅛th inch thick—more than twice as thick as most Motherboards.
- Totally shielded—all holes plated through; full bus terminations.
- Fits easily into any standard chassis.
- Masterite edge connectors—the finest quality connectors available.
- Reflowed solder circuitry.
- No soldering required.
- Designed for the S-100 bus.

The Motherboard price is: $150 (KIT)
$190 (ASSEMBLED)

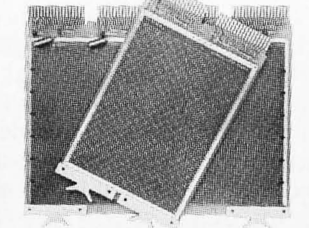
Five years of experience in every card

For five years, Artec has worked hard to develop a complete line of custom, prototype and off-the-shelf printed circuit boards. And in five years of tough industrial use, Artec boards have proven themselves among the most reliable boards available anywhere.

NEW! DEC® and Heath Compatible LSI Boards

The new Artec WW11 lets you adapt or add onto your DEC LSI-11 or Heathkit LSI minicomputer. Can accommodate 14 and 16 pin DIPs plus all necessary passive components.

FULL CARD $75
(10.45" x 8.4")
HALF CARD $35
(5.225" x 8.4")




® Trademark of Digital Equipment Corporation

Order today!

Put an Artec board to work for you. Use your Mastercharge or Visa. Or just send along a money order. We can accept only U.S. currency. Please include $3 handling on all orders. California residents add 6% sales tax.

Please send me: (include quantity)

___ Shielded Motherboard ___ Full WW11 Card ___ Half WW11 Card

☐ I've enclosed a money order for $ ______

☐ Mastercharge ☐ Visa ______ (number) ______ (exp. date)

Name ______

Address ______

City ______ State ______ Zip ______

Calif. Res. add 6% sales tax. Encl. $3 handling. 10% discount for students and computer club members. (Please enclose name of club or school)

ARTEC ELECTRONICS, INC.

Artec Electronics, Inc. • 605 Old County Rd. • San Carlos, CA 94070
(415) 592-2740

strings. Data of type character is very convenient in a microprocessor environment since a byte is the basic unit of memory.

The variable declaration section requires the naming of all identifiers that will be used as variables within this block. FORTRAN, BASIC, APL, and LISP do not adhere to this convention. Again PASCAL forces the user to think about what he wants to say before he says it. A sample variable declaration section might be

```
var    x,y:integer;
       cost:real;
       flag:boolean;
```

PASCAL's design allows the user to combine the utility of type declarations and variable declarations into data forms that would shame BASIC and FORTRAN. We have already seen PASCAL's predefined scalar variable types above. These are actually known as simple types.

Arrays may be multidimensional and include arrays of arrays.

Another simple type is the subrange type. Often a variable in a program may be expected to take on values only from a subrange of a simple type, say integers. For example

```
var asiz:1..100;
```

meaning "asiz" will be an integer whose values should lie between 1 and 100. Note that the compiler might choose to store "asiz" as a byte rather than a word if it was efficient enough to do so. Alternatively, if several variables are of the same range, a type statement could have been used

```
type  lsiz=1..100;
var asiz, bsiz, f1:lsiz;
```

Another simple type is the symbolic scalar type. This feature permits identifiers to be used in place of a sequence of integers, greatly enhancing the readability of the program. Suppose a program needed to represent the months of the year as a variable associated with some billing information. The approach in BASIC would be to use the sequence 1, 2, . . . , 12. PASCAL could use the subrange type 1 . . 12 or better

Items of different types may be aggregated into a single entity that can be stored as one logical unit.

```
type
   months = (jan, feb, mar, apr, may, jun,
             jul, aug, sep, oct, nov, dec);
   var billmonth,duemonth:months
```

In the statement section of a program, "billmonth" may be assigned one of the symbolic scalars from "months" or tested to see how its value compares with "duemonth." There are several functions available that operate on symbolic scalars, for example, **ord**(billmonth) would yield a number between 0 and 11 indicating the position of that month in the list "months."

Simple types are part of a more general data description called a type. Types include pointers which are used when dynamic data storage is referenced, file pointers which are used to reference secondary data storage, and arrays which are used with vector data storage. An example of an array declaration is

```
var cost: array[months] of real;
```

Notice that this array will be indexed, or subscripted, by "months." In general, arrays may be indexed by any simple types, may be multidimensional, and may be of any type, including arrays of arrays.

Two additional types set PASCAL in a class by itself; these notions allow powerful algorithm descriptions. The **set** type allows user manipulation of sets. Consider

```
var special: set of months;
```

The union, intersection, and set difference operators as well as relational operators may be applied to sets. A variable of scalar type may be tested for membership in a set of the same scalar type, for example

```
if billmonth in special then. . .
```

The last type is the record type. Items of different types may be aggregated into a single entity that can be stored as one logical unit, for example as one element of an array.

```
type
   customer = record
                name:array[1..20] of char;
                bal,bal30:real;
                datedue:daterec
              end;
   daterec =  record
                day:1..31;
                mo:months;
                year:integer
              end;
var
   database: array[1..100] of customer;
```

To reference fields of a record, the record name followed by a period, followed by the field name is used. Hence the over 30 day balance of customer 12 is "database[12].bal30" and the day of the due date of the current bill of customer 27 is "database[27].datedue.day." The full impact of record types cannot be explained in this short article; they must be used to be appreciated. One advantage of records is that items may be logically grouped together rather than stored in parallel arrays.

Procedure and function definitions would follow next in a program. They may be recursive and permit parameter passing in a style somewhat similar to ALGOL. Because of the position in a program of these declarations, procedures and functions may reference globally any variables or types defined in the main program. The body of a

procedure or function is identical to the body of a program; hence, procedures may be defined within procedures, and so on. Any variables defined within procedures or functions are considered local to the procedure and are unique to each invocation of the procedure. The sample program in listing 1 has several examples.

The statement portion of a program is called a "compound." A compound is a sequence of the keyword **begin**, any number of statements separated by semicolons, and the keyword **end**. The program ends with a period. Each of the statements within a compound may be one of a variety of different kinds of statements. Assignments, like

```
database[i+k].bal:=total
```

are the most common statements. PASCAL supports a large number of control statements which give the language its structure.

PASCAL has a looping control similar to that of standard BASIC but the step or increment may be only +1 or −1. The **for** statement causes a single statement, which could be quite complex, to be executed some number, including zero, times. For example

```
for ind. =1 to 100 do
   begin
      due:=1.006*database[ind].bal;
      database[ind].bal:=0.0;
      sum:=sum+due;
      database[ind].bal30:=
         1.006* database[ind].bal30+due
   end
```

This segment shifts the balance 30 days, adds some interest charge and accumulates a sum of the recently aged balances. If in a **for** statement, the increment were to be −1, then the keyword **downto** would replace the keyword **to**.

PASCAL supports both simple conditional and full conditional statements; that is

```
if <condition> then <statement>
         and
if <condition> then <statement>
                 else <statement>
```

Any "dangling **else**, an **else** which follows a sequence of "**if** . . . **then if** . . . then . . .," is paired with the innermost **if**.

When working with records, partial addressing can be done by using the "**with**" statement. This allows the fields of a record to be referenced as variables. The previous example then becomes

```
for ind:=1 to 100 do
   with database[ind] do
      begin
        due:=1.006*bal;
        bal:=0.0;
        sum:=sum+due;
        bal30:=1.006*bal30+due
      end
```

Three additional control statements are the **while**, **repeat**, and **case** statements. The **while** statement allows a given statement to be executed as long as some Boolean expression is true (the condition is tested first).

```
while <condition> do <statement>
```

The **repeat** statement allows one or more

Listing 1: The Polish "compiler" listing. Notice that PASCAL does not constrict the format of the program line. Indentation allows the program blocks to be easily separated from each other and makes the program easier to read.

```
PROGRAM PARSE(INPUT,OUTPUT);
(*PROGRAM PARSES SIMPLE ARITHMETIC EXPRESSIONS
  INTO THEIR RESPECTIVE POLISH CODE.  IT DOES
  THE PROPER TYPE CONVERSIONS NECESSARY FOR
  REAL AND INTEGER EXPRESSIONS ACCORDING TO
  THE FORTRAN CONVENTION:
         REAL: A-H, O-Z
         INTEGER: I-N
  VARIABLES ARE ONE LETTER LONG*)
LABEL 99; (*FOR ERROR RESTART*)
CONST
  DONTCARE='?'; (*MARKERS FOR CODE GENERATOR*)
  MAXPC=100;     (*MAXIMUM CODE SPACE*)
TYPE
  CODESPACE=1..MAXPC;  (*ADDRESS SPACE*)
  ATTR=(NONE,INT,REA); (*ATTRIBUTES OF OPCODES AND EXPRESSIONS*)
  LEXTY=(ADDOP,MULOP,LPAREN,RPAREN,IDENT,EOL);
         (*THESE LEXEMES FOR INPUT ASSUME A NON-HOSTILE USER*)
  INSTRUCTION=RECORD
                OPC:CHAR; (*OPCODE*)
                ITYPE:ATTR; (*OPCODE TYPE*)
                ADR:CHAR   (*NAME OF IDENT*)
              END;
VAR
  CODE:ARRAY[CODESPACE] OF INSTRUCTION; (*WHERE CODE GOES*)
  PC:CODESPACE; (*PC OF CURRENT INSTRUCTION*)
  GATTR:ATTR; (*GLOBAL TYPE OF EXPRESSIONS*)
  CH:CHAR;     (*CURRENT INPUT CHARACTER*)
  CHTYPE:ATTR; (*CURRENT CHARACTER ATTRIBUTE IF IDENT*)
  LEX:LEXTY;   (*LEXEME OF CURRENT INPUT*)
  BFR:PACKED ARRAY[1..80] OF CHAR; (*INPUT BUFFER*)
  BP:INTEGER; (*CHARACTER BUFFER POINTER*)
PROCEDURE SCAN; (*PROCESS NEXT INPUT CHARACTER*)
  BEGIN
    REPEAT
      BP:=BP+1;
      CH:=BFR[BP]
    UNTIL CH#' ';
    (*WORRY ABOUT END OF LINE*)
    IF ORD(CH)=0
      THEN LEX:=EOL
      ELSE
        IF CH IN ['A'..'Z']
          THEN
            BEGIN
              LEX:=IDENT;
              IF CH IN ['I'..'N']
                THEN CHTYPE:=INT
                ELSE CHTYPE:=REA
            END
          ELSE
            CASE CH OF
              '(':LEX:=LPAREN;
              ')':LEX:=RPAREN;
              '+','-':LEX:=ADDOP;
              '*','/':LEX:=MULOP
            END
  END (*OF SCAN*);
PROCEDURE ERROR;
  BEGIN
    WRITELN(' ':BP+1,'↑ ERROR'); (*COMPENSATE FOR USER PROMPT*)
    GOTO 99
  END (*OF ERROR*);
```

"Dynabyte builds confidence into my computer."

DYNABYTE DB8/2

Plug-in-and-run performance takes the uncertainty out of owning a computer. That's the kind of computer products Dynabyte builds.

Confidence in your computer is increased because you know that Dynabyte S-100 boards are factory assembled by trained technicians, burned in for 72 hours at elevated temperatures and continuously tested for three days.

That was true of the industry's first assembled and tested 16k dynamic RAM from Dynabyte, and it continues with all the other products from Dynabyte.

Once you plug Dynabyte boards in their careful design, factory prime components and quality construction keep them running. Dynabyte backs them with a one year warranty—the longest in the industry.

Incompatability is coming to an end. Dynabyte's 16k and 32k static RAMs come with access times of either 250ns for Z-80A processors or 450ns for 8080 and 8085 chips. You can use the 16k static RAMs with Alpha Micro or Cromemco bank switching memory expansion techniques.

Computer users are so confident of Dynabyte's 16k dynamic RAM they have made it the most widely used S-100 dynamic memory in the world.

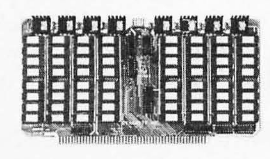
If you intend to become a proficient data processor but instead are unhappily debugging your system, or want to avoid a lot of debugging, then Dynabyte will be a great choice.

As you spend less time monkeying with hardware and more time computing, you'll become more confident of your abilities and proud of your accomplishments. Isn't that what owning a computer is all about?

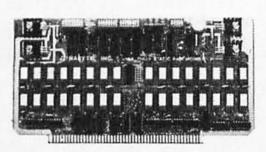
Tell your computer retailer with confidence that you want Dynabyte boards—the plug-in-and-run performers.

1005 Elwell Court,
Palo Alto, CA 94303.
(415) 965-1010.

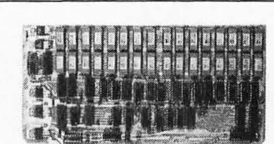
DYNABYTE



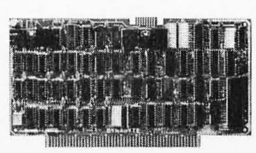
32k Fully Static RAM. 250ns or 450ns; 4k boundary addressing; no DMA restrictions; full Schmitt trigger buffering; no wait states; complete S-100 compatibility; conservative thermal design includes eight regulators and heat sinks. 250ns, $995. 450ns, $925.



16k Fully Static RAM. 250ns or 450ns; bank select is provided and is compatible with most popular schemes for memory expansion beyond 64k including Alpha Micro and Cromemco systems; 4k block addressing along 4k boundaries; write protect with alarm for each 4k block; full Schmitt trigger buffering. 250ns, $555. 450ns, $525.



16k Dynamic RAM. The industry's most popular S-100 dynamic RAM. Self-contained refresh logic is transparent to the 8080 processor and never generates a WAIT state. 1MHz direct memory access, 16k addressing. The Original Great Memory by Dynabyte, $399.



Naked Terminal. 80 character × 24 line = 1920 character display capacity. Not just a video interface but a full terminal; plug into a mainframe with keyboard and monitor and save! Upper and lower case; black/white or white/black; block mode editing; addressable cursor; half and full duplex; no software support required; addressable status and I/O ports. $350.

Circle 110 on inquiry card.

Listing 1, continued:

```
PROCEDURE GENCODE(F:CHAR; I:ATTR; A:CHAR);
  BEGIN PC:=PC+1;
        IF PC>MAXPC
          THEN BEGIN WRITELN('OVERFLOW');ERROR END;
        WITH CODE[PC] DO (*INDEX INSTRUCTION*)
          BEGIN OPC:=F;ITYPE:=I;ADR:=A END
  END (*OF GENCODE*);
PROCEDURE LISTCODE;
  VAR LPC:CODESPACE;
  BEGIN
    FOR LPC:=1 TO PC DO
      WITH CODE[LPC] DO BEGIN (*INDEX INSTRUCTION*)
         CASE OPC OF
           '+':WRITE('ADD');
           '-':WRITE('SUB');
           '@':WRITE('NEG');
           '*':WRITE('MUL');
           '/':WRITE('DIV');
           'F':WRITE('FLOAT');
           'P':WRITE('PUSH') END;
         IF OPC#'F'
           THEN
             BEGIN
               IF ITYPE=INT THEN WRITE('I') ELSE WRITE('R')
             END;
         IF OPC='P' THEN WRITELN(CHR(11B),ADR) ELSE WRITELN
    END (*OF WITH AND FOR*)
  END (*OF LISTCODE*);
PROCEDURE FIXUP(AX:CODESPACE;   (*PC OF FIX LOCATION OF OPERAND 1*)
                LOP:CHAR;       (*CURRENT OPERATOR*)
                LATTR:ATTR);    (*ATTRIBUTE OF OPERAND 2*)
  VAR TPC:CODESPACE;
  BEGIN
    IF GATTR#LATTR  (*TYPES DON'T AGREE*)
      THEN
        BEGIN
          IF GATTR=INT  (*FLOAT OPERAND 2*)
            THEN BEGIN GENCODE('F',NONE,DONTCARE);GATTR:=REA END
            ELSE (*HAVE TO FLOAT OPERAND 1, MOVE CODE UP*)
              BEGIN
                IF PC=MAXPC THEN BEGIN WRITELN('OVERFLOW');ERROR END;
                FOR TPC:=PC DOWNTO AX DO CODE[TPC+1]:=CODE[TPC];
                PC:=PC+1; (*TOOK ANOTHER WORD*)
                CODE[AX].OPC:='F'  (*FLOAT OPERAND 1*)
              END
        END;
    GENCODE(LOP,GATTR,DONTCARE) (*GENERATE OPERATION*)
  END (*OF FIXUP*);
PROCEDURE EXPR; (*HERE IS ALL THE WORK*)
  VAR
    LOP:CHAR;     (*CURRENT ADDOP*)
    LATTR:ATTR;   (*ATTRIBUTE OF OPERAND 2*)
    AXPC:CODESPACE;(*WHERE FLOAT OF OPERAND 1 GOES, IF NEEDED*)
  PROCEDURE TERM;
    VAR
      LOP:CHAR;   (*CURRENT MULOP*)
      LATTR:ATTR;(*ATTRIBUTE OF OPERAND 2*)
      AXPC:CODESPACE;(*WHERE FLOAT OF OPERAND 1 GOES, IF NEEDED*)
    PROCEDURE FACTOR;
      BEGIN
        IF LEX=IDENT  (*IDENTIFIER*)
          THEN
            BEGIN
              GATTR:=CHTYPE;
              GENCODE('P',GATTR,CH);
              SCAN
            END
          ELSE
            IF LEX=LPAREN
              THEN
                BEGIN
                  SCAN;EXPR;
                  IF LEX=RPAREN THEN SCAN ELSE ERROR
                END
              ELSE ERROR  (*JUNK INPUT*)
      END (*OF FACTOR*);
    BEGIN (*OF TERM*)
      FACTOR;
      WHILE LEX=MULOP DO
        BEGIN
          LATTR:=GATTR; LOP:=CH;
          AXPC:=PC+1; (*SAVE ADDR OF NEXT INSTRUCTION*)
          SCAN;FACTOR;
          FIXUP(AXPC,LOP,LATTR)
        END
    END (*OF TERM*);
  BEGIN (*OF EXPR*)
    IF LEX=ADDOP  (*LEADING SIGN*)
      THEN
        BEGIN
          LOP:=CH;SCAN;TERM;
          IF LOP='-' THEN GENCODE('@',GATTR,DONTCARE)
        END
      ELSE TERM;
    WHILE LEX=ADDOP DO
      BEGIN
        LATTR:=GATTR;LOP:=CH;
        AXPC:=PC+1; (*SAVE ADDR OF NEXT INSTRUCTION*)
        SCAN;TERM;
        FIXUP(AXPC,LOP,LATTR)
      END
  END (*OF EXPR*);
BEGIN (*OF MAIN PROGRAM*)
  WHILE TRUE DO (*INFINITE LOOP*)
    BEGIN
99:   REPEAT
        WRITE('>>'); (*PROMPT USER*)
        BP:=0; (*GET INPUT LINE*)
        WHILE NOT EOLN DO
          BEGIN
            BP:=BP+1;READ(BFR[BP])
          END;
        READLN (*RESET EOL INDICATOR*)
      UNTIL BP#1; (*GET A NON-EMPTY LINE*)
      BFR[BP]:=CHR(0); (* <NULL> FOR EOL*)
      PC:=0;BP:=0; (*SCAN FROM THE BEGINNING*)
      SCAN;
      EXPR; (*DOES ALL THE WORK*)
      IF LEX=EOL THEN LISTCODE ELSE ERROR
    END
END.
```

statements to be executed until a condition becomes true (the condition is tested last).

> **repeat** <statement> { ;
> <statement> } **until** <condition>

The brackets denote a portion that may occur zero or more times; for example

```
ind:=0;
repeat;
   ind:=ind+1
until (database[ind].bal>100.0) or
 (ind=100)
```

This will find the first customer whose balance is greater than $100, if one exists.

The **case** statement consists of an expression, known as the selector, and a list of statements, each labelled by one or more constants of the type of the selector. The statements whose constant is equal to the current value of the selector is executed. Some versions of PASCAL admit subranges for labels and an **else** or **otherwise** clause within a **case** statement.

```
case database[ind].datedue.mo of
   jan,feb,may:   <statement 1>;
   mar,jun,jul:   <statement 2>;
   oct,dec:       <statement 3>
end
```

Statement 1 will be executed if the due month is January, February, or May, and so on. Notice that no statement is executed if the month is April, August, September, or November. Of course, the nesting of such control statements is permissible and allows much more complex control structures to be implemented.

The **reset** and **rewrite** statements initialize input and output channels, respectively. Some versions of PASCAL do not require these for the default channels input and output. The IO commands are designed at two levels. To move primitive data to and from IO devices or files use the commands

Rackets or Racquets?

Software systems from TSC serve—whether your racket is business or pleasure. TSC's software is designed for business and industrial uses as well as for recreation. Off the job or at work, TSC software serves your needs.



Assembly Language Programs
(Includes Source Listings)

With Object Code Cassette

SL68-5C 6800 Space Voyage	$18.95
SL68-24C 6800 Text Editing System	$30.45
SL68-26C 6800 Mnemonic Assembler	$30.45
SL68-29C 6800 Text Processing System	$38.95

With Object Code Paper Tape

SL68-24P 6800 Text Editing System	$31.50
SL68-26P 6800 Mnemonic Assembler	$31.50
SL68-29P 6800 Text Processing System	$40.00
SL80-10P 8080 Text Editing System	$37.50
SL80-11P 8080 Text Processing System	$41.00

Cassettes are in the Kansas City Standard format. Many other programs are available. Send 25¢ for a complete catalog.



Technical Systems Consultants, Inc.
Box 2574
W. Lafayette, Indiana 47906
317-423-5465

Specialists in Software & Hardware for Industry & the Hobbyist

To Order: Include 3% postage, $1.00 handling on orders under $10.00, and Indiana residents add 4% sales tax. Check your dealer!

TSC Monthly Feature:

8080 Mnemonic Assembler

This is the most complete resident non-macro assembler available for the 8080 and the complete source listing is included! The assembler is fully Intel compatible except for logical expression operators. All standard pseudo-ops are supported plus features such as paging, titling, spacing, listing suppression, sorted symbol table, Intel format tape generation, hex or octal output, decimal line numbers, auto field formatting, hex, decimal, octal or binary constants, and more. 5.5K of RAM is required beginning at 1000 hex, plus source and symbol table space. An object code paper tape in Intel ASCII format is available.

SL80-12	**8080 Mnemonic Assembler**	**$25.00**
SL80-12P	**Assembler with paper tape**	**$34.00**

Circle 370 on inquiry card.

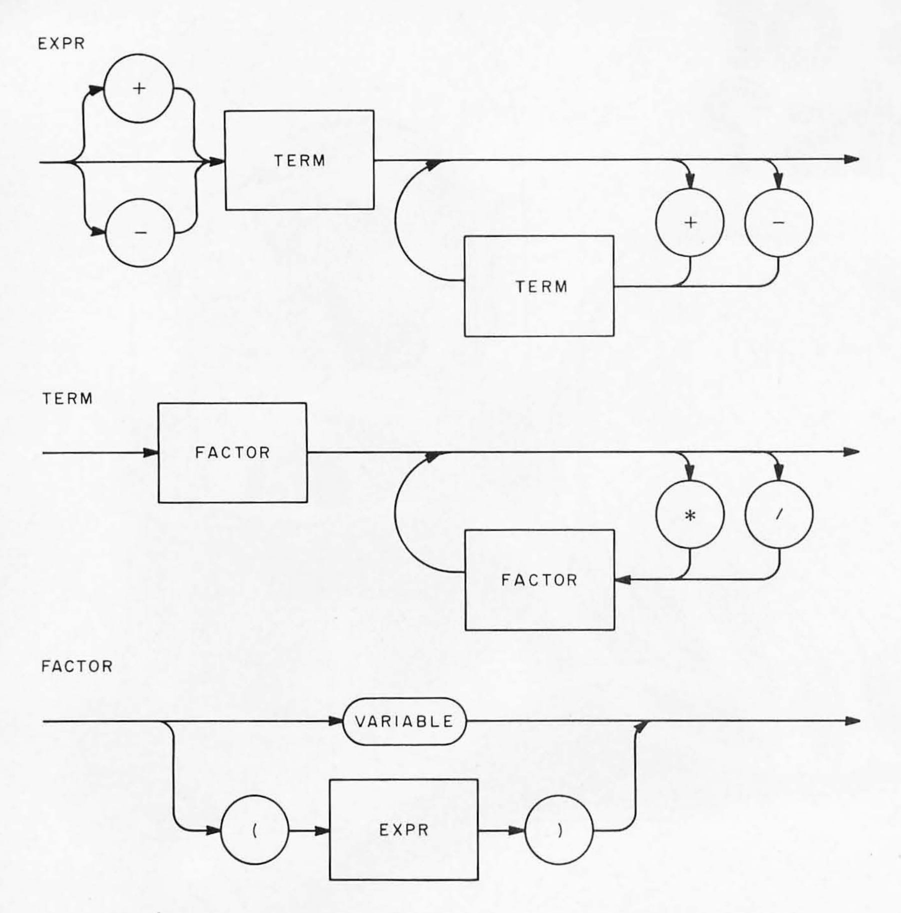


Figure 1: Syntax diagrams for generation of valid expressions. The diagram "expr" is entered from the left and calls term. Term calls "factor" which may call expr, etc. This model assumes that the only operations are addition, subtraction, multiplication and division.

An expression is an optional sign, a term, followed by any number of addition or subtraction operators and terms.

put or **get** respectively. To input or output an entire line or set of data we use **read**, **readln**, **write**, and **writeln** which are similar to FORTRAN IO commands. Formatting is done within the commands themselves. The **read** command will only input the necessary information (even if it must read several lines) while **readln** additionally discards the remainder of the current input line. The output commands, **write** and **writeln**, operate in an analogous fashion for output.

A significant example is now in order. Consider the problem of compiling an arithmetic expression. To greatly simplify the problem, assume all variables are one letter in length, no constants will appear, and the only operators will be +, –, *, and /. To make the problem interesting, assume that variables lettered a–h and o–z are of type **real** and the rest are of type **integer**. This is the same as the implicit types for FORTRAN. The program will produce code for a "stack machine." That is, the operators are applied only to operands already on the stack and the result will replace the operands on the stack. One task is the recognition of correct expressions. This may be done by several methods including precedence tables, LALR(1) parsers, and recursive descent. The latter will be used since it is the technique employed within most PASCAL compilers. Recursive descent compilation utilizes a set of recursive procedures to recognize its input, with no backtracking. To understand the algorithm, consider the series of "syntax diagrams" in figure 1.

To generate a valid expression, for example, one enters the diagram from the left, selects an arbitrary path through the diagram, and exits to the right. Any box encountered is to be treated like a subroutine or procedure call. A circle or box with rounded edges is to be the current input item. An expression is thus an optional sign, a term, followed by any number (including zero) of addition or subtraction operators and terms. Similarly, one can define a term. These definitions build in the normal precedence of operators and correctly handle a unary minus. Notice that <expr> will call <term>, <term> will call <factor> and maybe <factor> will call <expr> again. This would occur whenever parentheses were encountered.

A second task to accomplish is to properly handle the necessary type conversion of intermediate results. Many textbooks refer to this problem when discussing syntax directed translation but few illustrate "real" solutions. As an example (using the above assumptions) consider

```
J + K * X
```

It is not known that this expression must have a real value until the X is seen. The recursive descent phase, independent of type conversion might translate this to

```
PUSH J
PUSH K
PUSH X
MUL
ADD
```

for its equivalent Polish Notation: J K X * +. However, what is really required is

```
PUSHI J
FLOAT   (convert the top of the stack)
PUSHI K
FLOAT
PUSHR X
MULR
ADDR
```

where the operators have either "R" or "I" suffixed to indicate a real or integer operator, respectively. The suffix for the PUSH instruction is known as soon as the variable name is seen. The types for the arithmetic operators and the insertion of the FLOAT instructions must be added somewhat after both operands have been seen; in other

The PerSci generation of Diskette Mass Storage Systems Smarter, Faster, Smaller.

A new generation of advances in diskette drive technology comes together in PerSci Mass Storage Systems, the first diskette subsystems to provide large storage capacity, intelligence and high speed in as little as 4½" of space.

Two Times The Capacity...

New PerSci developments in large disk voice coil technology reduced to floppy disk applications make possible data densities as high as ½ Mbyte per diskette side—up to 2 Mbytes in a two drive system. When IBM data format compatibility is required, PerSci Systems will store ½ Mbyte per drive.

Microprocessor Intelligence...

PerSci's highly intelligent microprocessor based controllers, either single or double density versions, include the PerSci File Management Firmware. Under controller direction, the PerSci Mass Storage Systems are capable of performing many functions normally requiring CPU time and memory, including formatting, editing, reinitializing, automatic file and full diskette copy.

7 Times The Speed...

The unique voice coil positioning system of PerSci diskette drives allows PerSci systems to access data as much as 7 times faster than competitive systems. A random average seek is only 33 ms, as fast as many large disk systems.

In Half The Space.

PerSci Systems, one or two drive models, can be configured for mounting in a 19" rack or for table top operation. When maximum capacity in minimum space is called for, PerSci offers the "Slimline" system, only 4½" wide. A single or double density controller can be mounted in the standard system chassis or in the host. Interfaces to major microcomputers are available and an RS232 serial interface is optional.

Get all the most recent engineering advances in one smarter, faster, smaller system at competitive prices from PerSci, Inc., 12210 Nebraska Avenue, W. Los Angeles, Calif. 90025. (213) 820-3764.



Peripherals a Generation Ahead.

Circle 266 on inquiry card.

```
>>A+B
PUSHR     A
PUSHR     B
ADDR
>>A/I
PUSHR     A
PUSHI     I
FLOAT
DIVR
>>I/J
PUSHI     I
PUSHI     J
DIVI
>>J+K*X
PUSHI     J
FLOAT
PUSHI     K
FLOAT
PUSHR     X
MULR
ADDR
>>(I*J-(X+M))/(P+N)
PUSHI     I
PUSHI     J
MULI
FLOAT
PUSHR     X
PUSHI     M
FLOAT
ADDR
SUBR
PUSHR     P
PUSHI     N
FLOAT
ADDR
DIVR
>>A+B*
      ↑ ERROR
>>A*(B+I
         ↑ ERROR
>>I)*B
    ↑ ERROR
>>ZI
    ↑ ERROR
```

Listing 2: Sample program execution. After outputting a prompt the program waits for an expression to be input. It then lists all of the instructions that would be generated for a compiler code.

words, a fixup must be done. As one alternative, this may be accomplished by generating code in memory and keeping track of the type attribute of each operand and the addresses of where the last instruction for that operand was stored. If a type conversion is required on the first operand (of a binary operator), all code beyond the saved address is simply moved up one location and a FLOAT instruction is inserted. If a type conversion is required for the second operand, a FLOAT instruction is added as the last instruction in the evaluation of the second operand. *[In this paragraph and remaining text of the article, words in upper case refer to listing 1 . . .* RGAC*]*

The program in listing 1 is a solution to the expression evaluation problem. It is a direct implementation of the methods suggested. The main portion of the program is trivial; it asks for a line of input, calls procedure EXPR to parse the line, lists the output if there is no error, and repeats the process.

The type statements are important and quite varied. See that the constant MAXPC defines the maximum address space and is used in the declaration of the subrange type CODESPACE. The variables ATTR and LEXTY are symbolic scalar types and INSTRUCTION is a record type.

The variable CODE is an array of instructions. This is where the "compiled" code will reside. The type attribute of the second operand of an operation is stored in GATTR which is global to all the program's procedures.

The procedure SCAN picks up the next character(s), ignoring spaces and determines the correct token and type if it is a variable. Note the use of the case statement and the sequential nested conditionals.

The procedure ERROR outputs a line with an upward pointing arrow to indicate where the error occurred.

The procedures GENCODE and LISTCODE are responsible for encoding the instructions into the code array and decoding the code array for output respectively. The **with** statements simplify both the PASCAL and compiled codes.

Any discrepancy in types of operands is resolved by FIXUP which inserts the code for the operator itself. In a full compiler, FIXUP would also worry about strings and other data types and issue the appropriate error messages when needed.

EXPR does most of the work, together with the procedures TERM and FACTOR. They function exactly as described above. They are quite simple in appearance but function correctly as the sample runs illustrate. The symbolic scalars ADDOP and MULOP are quite useful in this design.

When properly segmented, any program should be similarly constructed and as easy to read or modify. A lot may be gained from using a top down design. Given the time, anyone could stretch this program into a full compiler whose output was a similar Polish code, and alternatively encode this program into their favorite assembly language. All the hard work has really been done in expressing the algorithm to solve the problem.

I heartily recommend that anyone seriously interested in PASCAL in particular and good programming style in general obtain the two books listed in the bibliography.■

BIBLIOGRAPHY

- Jensen, Kathleen and Wirth, Niklaus, *Pascal: User Manual and Report*, Springer Study Edition (2nd Edition), Springer-Verlag, 175 Fifth Av, New York NY 10010, 1975.
- Wirth, Niklaus, *Algorithms + Data Structures = Programs*, Prentice-Hall, Englewood Cliffs NJ 07632, 1976.

Consistency—or a Lack Thereof. . .

Notes by C Helmers

In this issue readers will note a lack of consistency in the typography of various articles on Pascal. One fairly obvious example of this is the use of all upper case and the use of normal English capitalization of the name of the language. Since Pascal is derived from a proper name, the proper notation is lower case with initial capitalization. For names which are acronyms, like FORTRAN, COBOL, PL/I, etc, the upper case notation is appropriate. While we strove for consistency within a given article, at least one article used upper case notation for the name Pascal, showing the extreme inertia of traditionalism at a point when it was too late to make the changes to final copy.

The second area of questionable typography is a bit more nebulous and less subject to editorial fiat when "camera ready" type is received from authors: the style of representation of Pascal program listings. The ideal style is of course that used by Niklaus Wirth in his book Algorithms+Data Structures= Programs, *published by Prentice-Hall in 1976. This style uses* bold face type *in lower case for representation of the Pascal language keywords. It uses italics for the representation of specific variable names, procedure names and literal values which are part of the program. In articles by authors Ken Bowles (page 122), Charles Forsyth and Randall Howard (page 50), Chip Weems (page 143), and Allan Schwartz (page 168) this notation was used. But in two of these cases, the authors supplied camera ready typeset copy along with the articles involved, in order to minimize potential errors due to keystroking. Since two of these were typeset at BYTE, and the other two were typeset with different type specifications on different machines, there is naturally a different aesthetic flavor to the listings in these articles. A close variant of this form is seen in the listings of David Mundie's article (page 110) where bold face type and normal type are mixed in the listing.*

There is yet another variation on the graphics used to represent Pascal programs, provided by the listings accompanying Stephen Alpert's article (page 78). Here, the camera ready listing was supplied by the author as printed on an upper case line printer, so keywords are indistinguishable from program details on the basis of typography alone.

*What can we conclude about this inconsistency? Our goal at BYTE from now on will be to asymptotically approach the notation of Pascal programs in the bold face and italic form whenever we do the actual typesetting of a listing. The italic and the bold face typography provides an excellent contrast to normal type when elements of a program are mentioned within text. But when a manuscript comes with a usable camera ready listing of a Pascal program, such details of aesthetics must take second place to the goal of minimizing errors of transcription: it is far better to use a camera ready image derived from a machine produced listing than to key in a program manually in order to create a typeset form of the listing. Like programming, execution of a magazine production task once a month is fraught with myriad details. . .*CH

All figures and photos courtesy of The Franklin Institute, Philadelphia PA 19103.

Philadelphia's 179 Year Old Android

Charles F Penniman
The Franklin Institute
Philadelphia PA 19103



Photo 1: The 179 year old Maillardet writing and drawing automaton on display at The Franklin Institute in Philadelphia.

Cuckoo clocks, computers and dolls with rolling eyes somehow fascinate us all. The fascination seems to stem from our delight that people can make contraptions which do things by contrivance that are usually done by living men and beasts. But whatever the reason for it, we find animated statues in ancient China and in the temples of classical Greece. In Europe, the clockmakers of the Renaissance often adorned their works with marvelous moving figures. The famous tower clocks of Berne and Messina and the remarkable clock in the Cathedral at Strasbourg are just a few examples.

For us who live toward the beginning of the Electronic Age, it is hard to imagine the excitement that existed in the early years of mechanism. The automaton at the Franklin Institute that writes poems and draws pictures dates from those times. In the same way that they made machines to perform marvelous and delightful things, we program computers and build microprocessors to perform even more amazing feats. It is much the same phenomenon.

The Franklin Institute's mechanical lady dressed in green is one of the most important of the small number of androids that have ever been built with the ability to write and draw. The first machine with such capabilities was built around 1750 by Friedrich von Knauss working in Germany, but it was from Pierre Jaquet-Droz, Jean-Frederic

Leschot, and a succession of their collaborators that the most elegant machines came. In 1774 they produced their first writing doll in Neuchatel, Switzerland. The machine now at The Franklin Institute in Philadelphia was built about 1805 in London by Henri Maillardet, an associate of Leschot and Jaquet-Droz. His automaton is particularly distinguished by its unusually large memory and excellent movements.

The Maillardet machine weighs about 250 pounds (113.4 kg) and consists of a figure kneeling at a writing desk mounted atop an ornate stand containing the program and driving mechanism. Information for the doll's movements is communicated up through the body of the figure by an incredibly intricate combination of levers, rods, pulleys and cams.

The heart of the writing and drawing operation is actually a mechanical "read only memory" in the form of an array of disk cams rotating on a common shaft to drive the right hand of the figure. The cams are driven by a spring motor located at one end of the base that is coordinated with a second motor located at the other end of the base. This motor is used to slide the stack of operating cams transversely on their shaft into the proper position to produce the desired readout. The information contained in the undulations of the selected set of cam surfaces is picked up by three cam followers linked to the doll's hand to produce the required right and left, up and down, and vertical motions. There are seven programmed designs from which to choose: two poems in French, one in English, and three graceful pictures. Two designs require four sets of three cams each; the remaining designs are each on three sets of cams. This adds up to a total of 96 operating cams to govern the motions of the right hand of the figure. Additional, and far simpler, cams move the left hand, head and eyes.

The machine is marvelously complex, but perhaps the greatest marvel is that it can still function after nearly 175 years. Apparently very little wear has taken place. The details of the drawings are still remarkably sharp and the writing quite legible. The complex of linkages between the rotating cams and the motions of the hand still operate with no detectable play or slop in the bearings: a considerable achievement for engineering and technology. How this was achieved is largely a matter of conjecture today, but the automaton was built in an age in which trade secrets were kept closely within the circle of one's apprentices and family. It would be interesting, for instance, to know exactly how the machine was programmed. One can speculate that the profiles of the cams were



Figure 1: One of the incredible drawings executed by Maillardet's writing and drawing automaton in approximately five minutes. The automaton is currently on display at The Franklin Institute.

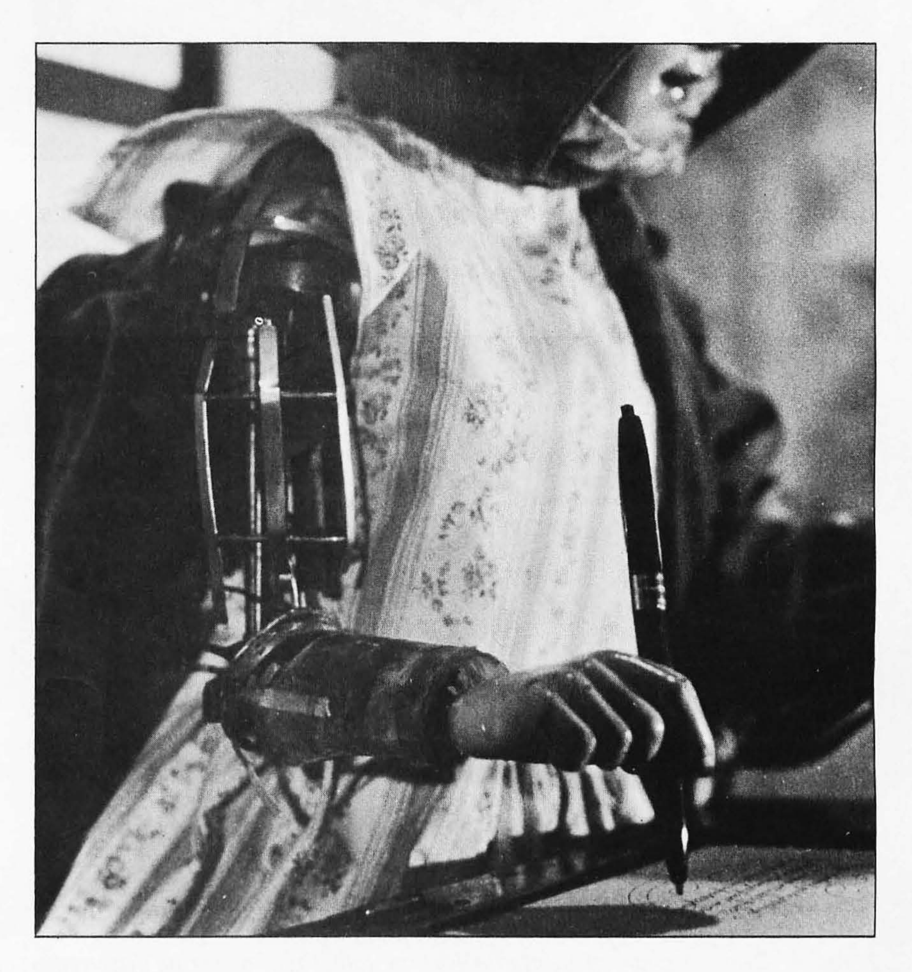


Photo 2: Right arm of the automaton, undraped. Modern ballpoint pen is not historically correct, but is effective.

laid out after the doll was constructed by moving its hand over a master drawing and tracing the corresponding motions of the three cam followers on simultaneously rotating disks of brass which were then cut and filed to their proper shapes. Yet, this is only a guess. The only thing that we know for certain in this regard is that the profiling of the cams had to be done with the greatest care and precision since there is up to a tenfold magnification of any possible error due to the multiplying effect of the linkage between the cam followers and the writing instrument in the doll's hand.

Some of the elaborate and delicate mechanisms that Maillardet made were sold to the wealthy. Occasionally they were comissioned as state gifts. In fact, the only other doll with the ability to write that can be attributed directly to Henri Maillardet was made for King George III to give to the Emperor of China during a period when the English were attempting to establish favorable trade relations with that country. That machine was programmed to write, in Chinese, flattering messages to the Emperor. It made the trip to China successfully and is reported to be alive and well in a museum in Peking. However, most of the automata that were built, including the one at The Franklin Institute, went into show business and toured all over Europe. Surviving advertisements attest to their popularity. Through newspaper clippings, the progress of the Philadelphia machine can be traced from France to Russia and throughout England until 1850. It is possible, but not altogether certain, that it was purchased at about that time by the great American showman, P T Barnum, for his American Museum. By some process, now unclear, it came to be owned by a Philadelphian, John Penn Brock, whose grandchildren donated it to The Franklin Institute. Perhaps it was the fire that destroyed the museum Barnum set up in Philadelphia that damaged the machine, but it was indeed a charred mass of wreckage when it was delivered to The Franklin Institute in November of 1928. Although Maillardet had his automaton originally fitted out as a little boy in court dress, by the time it came to The Institute, the costume had been changed to that of a French soldier. At The Institute, the machine was stored in one place and then in another until a staff machinist, Charles Roberts, became interested in trying to repair it. He was tremendously proud of his success in doing so as, indeed, he should have been. New clothes were made, but this time the doll was put into a dress instead of a boy's suit. The restoration of the original motion of dipping a pen (or perhaps it was a brush) into an inkwell turned out to be impossible. Roberts substituted a stylograph pen which has since been replaced by a totally unhistorical, but much more convenient, ballpoint pen. And, of course, it was necessary to make a number of new parts, but the only significant alterations made were to the writing instrument and to the sex of the doll.

Tradition in the Brock family had it that the automaton had been built by Maelzel, a considerable showman, the inventor of a metronome, and the builder of a number of

Photo 3: Rear view of the automaton. Cam motion is carried through the body of the doll and across the shoulders to the right arm by a system of rods, levers and shafts.

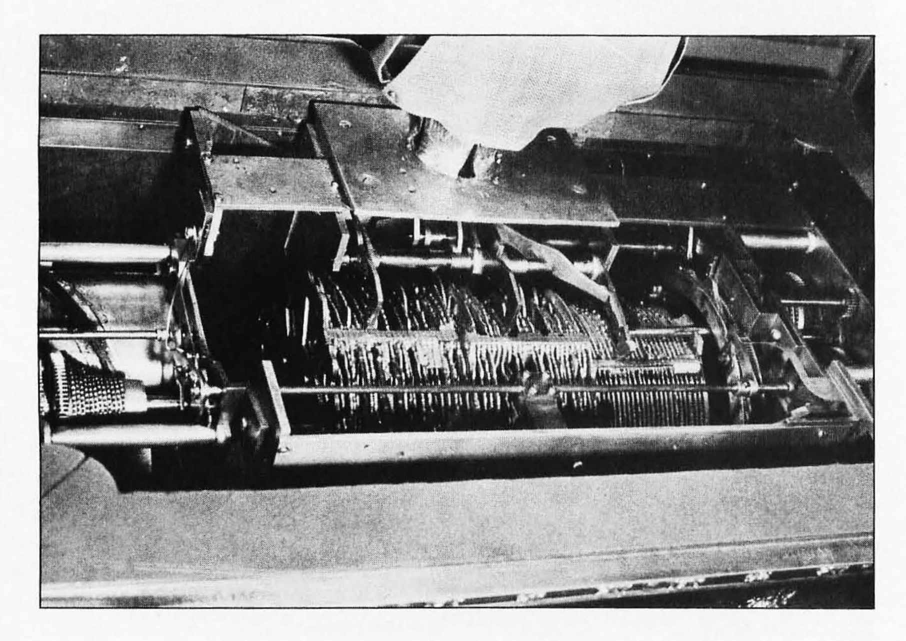


Photo 4: Drive mechanism located below the tabletop. The spring driven motor for the operating cams is at right. The X, Y and Z cams with the three followers are in the center. To the left is the program selection drive motor.

___ **Microcomputer Problem Solving Using PASCAL** by Kenneth L Bowles. This book is designed both for introductory courses in computer problem solving at the freshman and sophomore college level, and for individual self-study. Graphics is stressed in this version of the book, in many cases borrowing from the "Turtle Graphics" approach originated by Seymour Papert of MIT. A complete single-user software system based on PASCAL has been developed at the University of California at San Diego, where the author is a professor in the Department of Applied Physics and Information Science. This system embodies extensions to the standard PASCAL which include the necessary functions and procedures for handling graphics and strings. 563 pp. $9.80.

___ **An Introduction to Programming and Problem Solving With PASCAL** by G M Schneider, S Weingart, and D Perlman. This book has three major goals:

(1) To introduce *all* aspects of the programming and problem solving process, including problem specification and organization, algorithms, coding, debugging, testing, documentation, and maintenance.

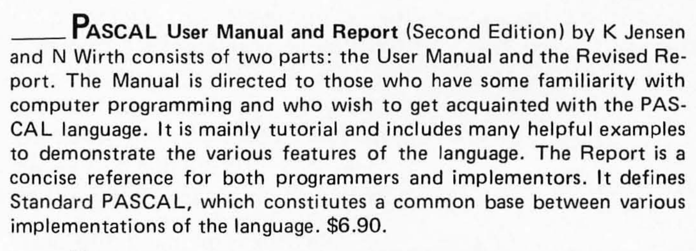
(2) To teach good programming style and how to produce a high quality finished product. This is brought out in numerous style examples throughout the text.

(3) To teach the syntax of the PASCAL programming language.

PASCAL is used as a vehicle to teach various aspects of programming techniques. $12.95.



Speaking of
PASCAL . . .



___ **PASCAL User Manual and Report** (Second Edition) by K Jensen and N Wirth consists of two parts: the User Manual and the Revised Report. The Manual is directed to those who have some familiarity with computer programming and who wish to get acquainted with the PASCAL language. It is mainly tutorial and includes many helpful examples to demonstrate the various features of the language. The Report is a concise reference for both programmers and implementors. It defines Standard PASCAL, which constitutes a common base between various implementations of the language. $6.90.

___ **Programming in PASCAL** by Peter Grogono. This book is an excellent introduction to one of the fastest growing programming languages. The text is arranged as a tutorial containing both examples and exercises to increase reader proficiency in PASCAL. Besides sections on procedures and files, there is a chapter on dynamic data structures such as trees and linked lists. These concepts are put to use in an example bus service simulation. Other examples range from the Tower of Hanoi problem to circumscribing a circle about a triangle. **Programming in PASCAL** is sure to hold the reader's interest. 359 pp. $9.95.

___ **The Design of Well-structured and Correct Programs** by S Alagic and M A Arbib. This book represents ten years of research in top-down program design and verification of program correctness, and demonstrates how these techniques can be used in day-to-day programming with PASCAL. An explanation of control and data structures and many examples of programs and proof development are provided. As a programming text, this book contains an introduction to the language, provides algorithms which operate on sophisticated data structures, and offers the full axiomatic definition of PASCAL in terms of proof rules. To use this book, no particular mathematical background is necessary beyond the basic idea of a mathematical proof, although an introductory course in programming is required. 292 pp. $12.80.

BITS T.M.

The Microcomputer Bookstore

25 Route 101 West
Peterborough, NH 03458

DIAL YOUR BANK CARD ORDERS ON OUR **TOLL-FREE HOT LINE: 1-800-258-5477.**

Name ________________

Address ________________

City ________ State ________ Zip ________

Signature ________________

☐ My check enclosed:

Check # : ________________

Amount: ________________

☐ My card # : ________________

Expires: ________________

		Total
Number of books	________	________
Postage $.75/book or $1.00 outside U.S.		________
Grand Total		________

Prices subject to change without notice. *Dealer Inquiries Invited.* *You may photocopy this page.*

Circle 32 on inquiry card.

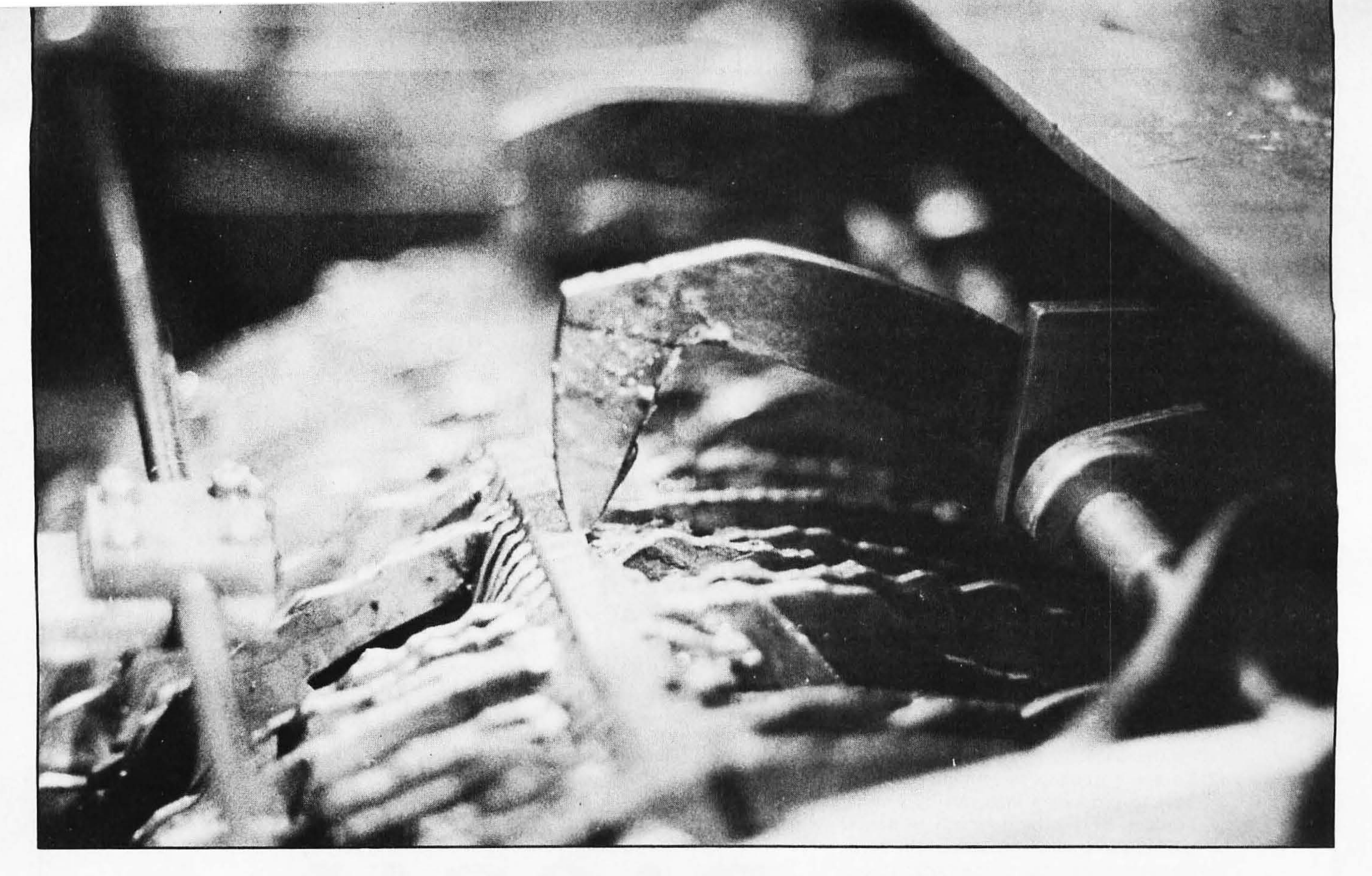


Photo 5: Center follower at rest on the bar on which it slides as the cam deck is positioned laterally. The undulations of the cams are also visible.

automata. In his memoirs, P T Barnum records the purchase of several automata from him. But after being repaired, the automaton herself set things right when her memory was read out. Following the last line of her last poem, the hand continued to write in its clear but quaint style: *"Ecrit par L'Automate de Maillardet,"* meaning "Written by Maillardet's Automaton." With this clue, locked for nearly 90 years in the memory of the machine, it was possible to search out and determine its proper origin.

The one English poem that she knows how to write (see page 102) is as follows:

Unerring is my hand, tho' small.
May I not add with truth:
I do my best to please you all;
Encourage, then, my youth.

Certainly her hand cannot be as unerring as it was in 1805. It would be interesting to have a sample of her writing from that time to make a comparison. But she still does her best to please and amaze us.■

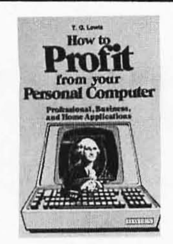
Enfant chéri des dames,
Je suis, en tout pays,
Fort bien avec les femmes,
Même avec les maris.
Ecrit par L'Automate de Maillardet

Figure 2: An example of the written output of Maillardet's writing and drawing automaton. The French poem is freely translated as: "A child by ladies adored,/I am throughout all lands,/In good favor with women,/And also their husbands." Below the poem in French is the inscription "Written by Maillardet's Automaton."

You May Have Seen Her in Action . . .

In an excellent WGBH NOVA *presentation on* "Artificial Intelligence" *aired on public television stations in March 1978, The Franklin Institute automaton by Maillardet may have been seen in action by many readers. (Also seen in action were robots NEWT and Shakey of contemporary vintage. The program also featured interviews with science fiction writer Arthur C Clarke, and a number of artificial intelligence researchers regarding the prospects for the near and far future of smart machine technology.)*

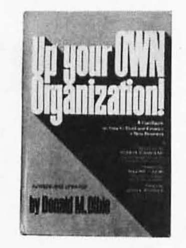
Get Down to Business



How to Profit From your Personal Computer

by Ted G Lewis

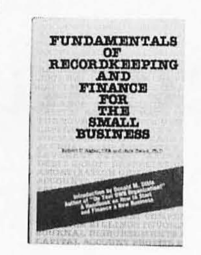
___If you want to use your personal computer in a profitable manner, and you need some help planning programming techniques this book is for you. It contains suggestions for accounting, payroll handling, inventory management, and sorting mailing lists. Many terms and notations are explained. Sample programs in BASIC, the use of blueprints to design program structure, and a full glossary of terms are a few of this book's special features. 191 pp. $7.95.



Up your OWN Organization!

by Donald M Dible

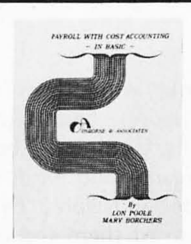
___A great handbook on how to start and finance a new business, this is the most comprehensive reference we've seen on the subject. For the programmer-consultant or the basement homebrewer-turned-entrepreneur, this is your book. It is recommended in the Bank of America *Small Business Reporter* and *Changing Times* magazines. 372 pp. Available for $14.95 in hardcover.



Fundamentals of Recordkeeping and Finance for the Small Business

by Robert C Ragan, CPA, and Jack Zwick, Ph.D

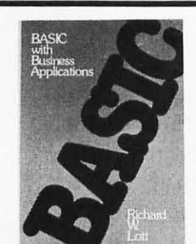
___Once you have your organization or business up and running, records must be kept. What should I keep, and how do I record them? This book on fundamentals will give you a helpful start. Section one deals with maintaining records, protecting assets, and providing a basis for planning. Section two provides a starting point for owner-managers wanting to sharpen their financial management skills. 196 pp. $10.00.



Payroll With Cost Accounting in BASIC

by Lon Poole and Mary Borchers

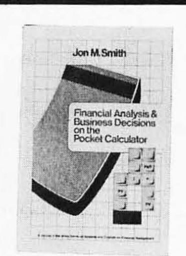
___This book includes program listings with remarks, descriptions, discussion of the principles of each program, file layouts, and a complete user's manual with step-by-step instructions, flowcharts and sample reports with CRT displays. All 35 programs are written in the widely used computer language BASIC, and work together to produce a payroll, right down to the printing of paychecks and maintaining of employee records. $12.50.



BASIC with Business Applications

by Richard W Lott

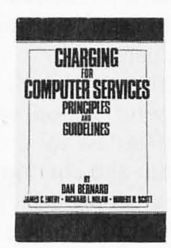
___This book focuses on the BASIC language and its application to specific business problems. The book is divided into two sections. Part one introduces the BASIC language and the concept of logical flowcharting. Part two presents problems and possible solutions. Topics include: interest rate calculation, break-even analysis, loan rates, and depreciation. Exercises at the end of each chapter give a greater understanding of BASIC by actual programming. $10.50.



Financial Analysis and Business Decisions on the Pocket Calculator

by Jon M Smith

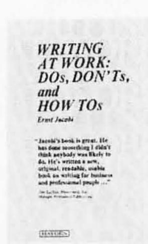
___This book is designed to aid the experimenter in performing applied analysis. It gives a variety of numerical techniques, approximations, tables, graphs, and flowcharts for calculations. All methods have been optimized for the pocket calculator. Topics include: calculating present and future values, consumer finances, real estate calculations, business statistics, and systems analysis. 317 pp. $14.95 hardcover.



Charging for Computer Services

by D Bernard, J C Emery, R L Nolan, and R H Scott

___This book is written for managers who must deal with service charges. This book provides the manager with principles and guidelines for a better understanding of the charge problem. The book provides general design principles along with specific suggestions to deal with specific problem areas. **Charging for Computer Services** is a necessary book for the manager making decisions in this vital area. 120 pp. $10.00, hardcover.

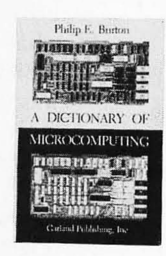


* NEW!

Writing at Work: Do's Don't's, and How To's

by Ernst Jacobi

___This is not a grammar or usage text. The goal of this informative book is "to turn writers into communicators." Addressed to the educated business or professional person who can write adequately, this guide is full of practical advice that will help you make your writing enjoyable, effective, and successful. 198 pp. $7.95.



A Dictionary of Microcomputing

by Philip E Burton

___In the opinion of BYTE's editor, Carl Helmers, "This is one of the best designed and executed dictionaries of computer related terms yet seen on the market. It is of particular relevance to those individuals who want a good general reference to numerous technical terms, broadly covering hardware and software fields as currently practiced." $12.50 in hardcover.

For your convenience in ordering, please use this page plus the order form on page 93. You may photocopy this page. **DIAL YOUR CHARGE CARD ORDERS TOLL-FREE 1-800-258-5477.**

BITST.M. **The Microcomputer Bookstore**

25 Route 101 West, Peterborough, NH 03458

Circle 33 on inquiry card.

Antique Mechanical Computers

Part 2: 18th and 19th Century Mechanical Marvels

Dr James M Williams
58 Trumbull St
New Haven CT 06510

In "Part 1: Early Automata," page 48, July 1978 BYTE, we traced the development of antique mechanical computers up to the middle of the 18th century, and described such devices as Vaucanson's mechanical duck. Now we continue with a discussion of talking, writing and music playing automata of the 18th and 19th centuries. (The discussion is not meant to be an exhaustive one, of course, since that would be beyond the scope of this series.)

Later Automata

Vaucanson's creations blazed across the scene in Europe 240 years ago, casting new light into hitherto dark places by showing what the dedicated mechanician could achieve. But, even after Vaucanson, the way was difficult. 38 years passed before a second flute playing machine was seen, a seated pair of rustics built by Duchamps in 1776 and said to be capable of playing 13 tunes. 109 years after Vaucanson made the original mechanical duck, a mechanician named Rechsteiner, who had restored that original duck, produced and displayed a duck of his own. Rechsteiner's duck was the product of three years of work. It appeared in 1847 and was the last automaton animal of note.

In the last quarter of the 18th century first a few, then a flood of automata began to appear.

In the last quarter of the 18th century, first a few, then nearly a flood of automata began to appear, as clockmakers began to realize not only the possibilities of their craft but also the splendid prices their premier work might command. The more standard automata such as ornamented clocks, from snuffbox size to prodigies bigger than steamer trunks, with processions of moving allegorical figures, spirals, pinwheels, and waterfalls, chimes, bells, dulcimers, whistles, organs, and birdcalls, continued to be made and sold. Every titled person had a score of them and men of substance could own several. The clockmaker of ambition knew where his challenge lay. There were mysteries to be created in machinery, and money and fame to be had. Mechanicians began to devote themselves to duplicating the physical action of parts of the human body. They chose part-behavior because of the immense difficulty of fabricating a mechanism that could imitate even *one* of the coordinated acts humans orchestrate into the continuous chain of actions; namely, behavior.

It is worth noting that in adults the discrete units of purposeful action which seem so integrated and effortless to most of us are anything but smooth and coordinated in early childhood. Most people can recall their clumsiness and exasperation in learning to tie their shoes or button their garments. The most intense concentration and dedicated repetition is required to cause these action patterns to set in our central computing mechanism (see "The Brains of Men and Machines," parts 1, 2, 3 and 4, February thru April 1978 BYTE), but once the setting (ie: learning) takes place over time, it becomes possible for us to execute one of these unit actions at will, devoid of effort and concentration. (The mechanism and locus of the setting is obscure: so is other memory storage. Lately, the cerebellar complex is viewed as the best candidate for unitary motor

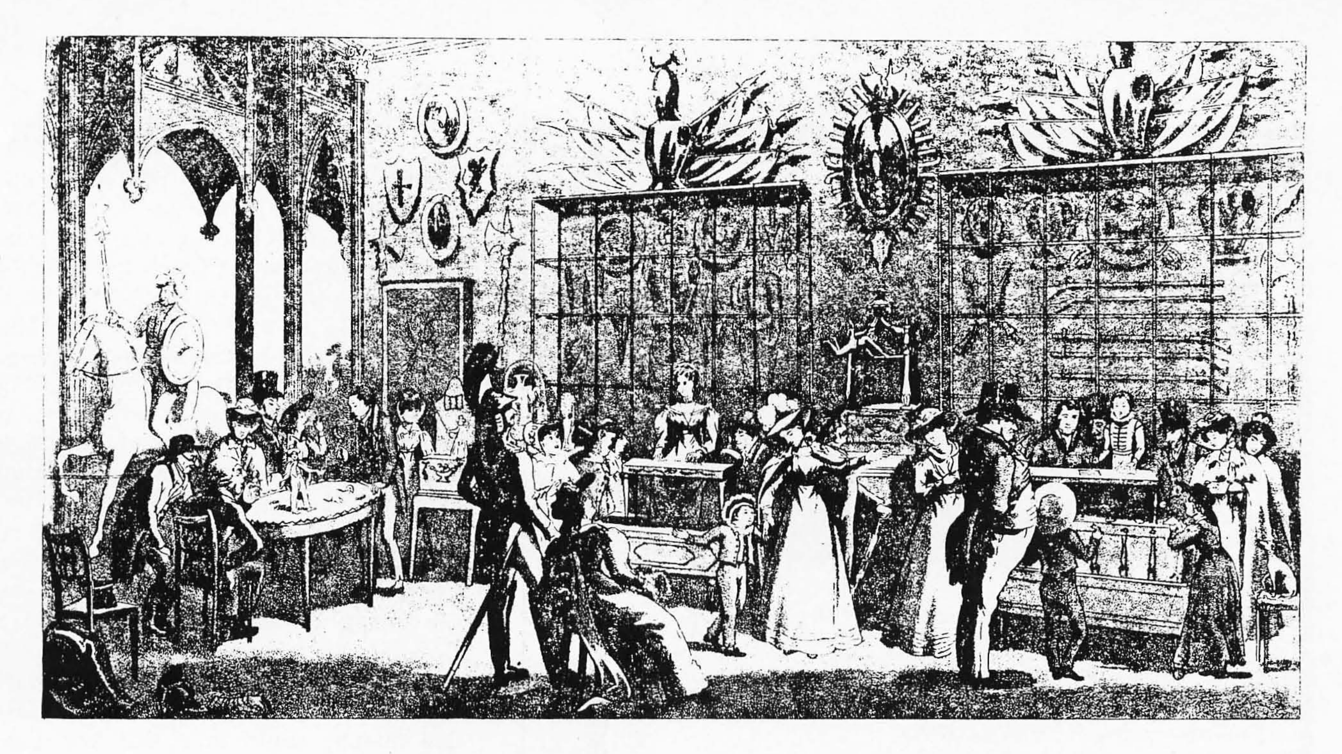


Figure 1: Aquatint etching of the Automaton Exhibition held at Gothic Hall in London in 1836. Various automata are shown; the one at the far right is evidently Jacquet-Droz's writing and drawing automaton now in the collection of the Franklin Institute in Philadelphia (see "Philadelphia's 172 Year Old Android" by Charles F Penniman, page 90 in this issue). The figure is shown dressed as a boy, but women's clothes were substituted when the unit was rebuilt in 1936 at the Institute. Exhibitions like this were relatively common in the 19th century. Engraving courtesy Charles F Penniman.

actions.) We can tie shoelaces behind our backs, a thing we never practiced or learned. Even extreme situations, like tying shoelaces while wearing mittens or hanging by the knees from a trapeze, do not begin to strain the capacities of our interior computing mechanism. The required actions have been "frozen" into our brains. Not only are they refractory to disarrangement (they endure as long as we live) but they are also flexible enough to permit our adapting them to novel circumstances. We all possess within us many thousands of such unitary chunks of learned behavior, now fully automatized and playable on command.

This is the part-behavior, smooth, continuous and automatic, that was being imitated by mechanicians. It requires substantial storage of program to duplicate. From our vantage point program storage is the most important feature these machines possessed. Consequently, many very beautiful mechanisms (the display pieces of Carl Faberge, jeweler to the Imperial Russian Court; a wide range of novelties such as soothsayers, magicians and other conjurers, acrobats and ropewalkers, agile harlequins and jugglers, automatic confectioners and wine stewards, and a great many more display mechanisms) are not mentioned here because they had little stored programming.

Walking and Running Machines

Early walking and running automata were represented only by dolls and toys. They were essentially trivial, programmed devices for they always very ingeniously arranged an *apparent* walking action (only a simple repetitive motion). The walk lacked directionality, nor was there provision for walking on other than smooth surfaces. It would be difficult to design a machine to walk in the same sense that people do: that is, the weight of the trunk is for a moment supported by one leg alone while the other leg is being drawn forward for a next step. Walking is in fact organized falling, with the mobile extremity brought forward just in time to forestall disaster. When you stop to recall that every known mechanical man actually rolls on wheels, and that at least three wheels are always employed to define the plane, you gain a new respect for human locomotion and a valuable perspective on the limitations of mechanisms that undertake to imitate it.

How can one describe machines so marvelously devised and tutored in their tasks that they rival the actions of human beings?

Speaking Machines

As far as I can discover, no programmable device uttering words, or their approximations, was ever known before the late

19th century (or even in later periods up to the time of Bell Labs' *Vodor* of 1939 World's Fair fame, which required an operator). Still, some remarkable devices appear to have existed. Leaving aside the brazen talking heads that dot Greek and Byzantine mythology (they were without a doubt all hoaxes), we learn that the Abbe' Mical in 1774 was said to have exhibited two talking heads which he later destroyed. In 1779 Kratzenstein won a prize offered by the Russian Imperial Academy of Science for a device that could pronounce distinguishable vowels. This device was made from a set of five specially shaped pipes. Baron Wolfgang Kempelen, creator of the Great Chess Automaton, worked for many years on talking devices, and one was said by Goethe to be ". . . able to say some childish words very nicely." The machine was a kind of bellows, soundbox, artificial tongue and mouth contrivance that the Baron manipulated under cover of a cloth; it now resides in a museum in Munich. Farber invented a machine which apparently spoke well enough to induce P T Barnum to purchase it for exhibition, in 1873. The device was operated by a keyboard.

It is a very curious thing that investigation of artificial speaking devices was so neglected by gifted mechanicians, for speech is the unique achievement of man. Moreover, the ear is so adaptable and forgiving of faults in the spoken word that virtually any kind of squawk might pass for a sentence. The mechanical problems would have been very great, but not insuperable.

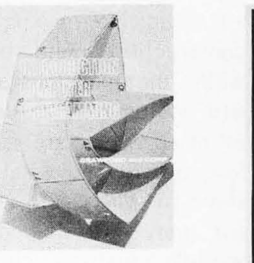
Writing Machines

Between 1753 and 1760 Friedrich von Knaus of Darmstadt devised and constructed four different machines that wrote block letters or cursive script according to programming using a quill pen and ink with programmed pauses to dip the pen. One machine produced three texts from three pens, while the last machine could inscribe up to 107 letters of preset text from its stored program or write individual letters one at a time from dictation under control of the operator. It may accurately be described as the first typewriter or scriptwriter. The mechanism appears to have been a cluster of shaped cams on which rode an array of cam followers, each one directing movements of the pen to form a letter. Text composition was managed by a drum that bore many rows of holes into which studs could be placed to activate the required cam. Thus text was easily altered by changing the pattern of studs. The tablet,

A Bit of the BASIC

---**Computer Resource Book—Algebra** by Thomas A Dwyer and Margot Critchfield is an exciting new way to learn about algebra and the interesting things you can do with it using a computer. The book uses the BASIC language, and flowcharts are used throughout to show the structure of programs. There are 60 applications programs including straight line graphs, polynomial equations, a space probe navigator, temperature profiles, computer generated animation, the ultramatic root finder, random number generation and many more. Although it is particularly suitable for students, just about everyone will find some intriguing and easy to use applications in this entertaining book. $4.80.

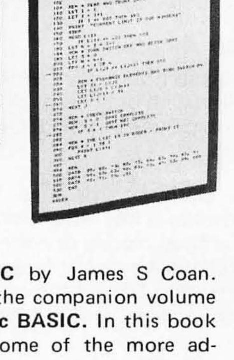




---**Introduction to Computer Programming** by Rudd A Crawford Jr and David H Copp. Here is an excellent way to learn about the general aspects of computer programming. **Introduction to Computer Programming** makes use of a hypothetical computer model and set of assembly language instructions designed to help the beginner see what goes on in computer programs. The emphasis throughout is on general principles; such concepts as loops, decisions, flowcharts and IO routines are covered in detail. The book also provides many example problems and prompts the reader by posing several quiz questions. Anyone who masters its contents will have a solid foundation for the study of practical assembly and high level languages. It is especially recommended for students, but just about everyone new to the subject should profit from it. $4.35.

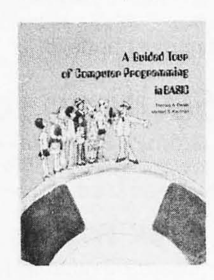
---**Basic BASIC** by James S Coan. If you're not already familiar with BASIC, James Coan's **Basic BASIC** is one of the best ways to learn about this popular computer language. BASIC (which stands for Beginner's All-purpose Symbolic Instruction Code) is easy to learn and easy to apply to many problems. **Basic BASIC** gives you step-by-step instructions for using a terminal, writing programs, using loops and lists, solving mathematical problems, understanding matrices and more. The book contains a wealth of illustrations and example programs, and is suitable for beginners at many different levels. It makes a fine reference for the experienced programmer, too. $7.95.





---**Advanced BASIC** by James S Coan. **Advanced BASIC** is the companion volume to James Coan's **Basic BASIC.** In this book you'll learn about some of the more advanced techniques for programming in BASIC, including string manipulation, the use of files, plotting on a terminal, simulation and games, advanced mathematical applications and more. Many useful algorithms are covered, including some clever sorting techniques designed to reduce program execution time. As with **Basic BASIC,** there are many illustrative example programs included. BASIC doesn't have to be basic with **Advanced BASIC**! $6.95.

---**A Guided Tour of Computer Programming in BASIC** by Thomas A Dwyer and Michael S Kaufman. Colorful graphics abound in this lively introduction to the BASIC language. The authors have tried to present a rigorous, yet entertaining approach to the subject. Written for the novice, **A Guided Tour** begins with a section on how to recognize a computer, followed by some tips on working at a terminal. By the end of the book readers are writing their own programs and solving elementary problems in finance and business. The emphasis throughout is on learning by doing. Anyone interested in computer programming should benefit from **A Guided Tour of Computer Programming in BASIC.** $4.80.





---**Some Common BASIC Programs** by Lon Poole and Mary Borchers, published by Adam Osborne and Associates. At last, a single source for all those hard to find mathematics programs! **Some Common BASIC Programs** combines a diversity of practical algorithms in one book: matrix multiplication, regression analysis, principal on a loan, integration by Simpson's rule, roots of equations, operations on two vectors, chi-square test, check writer, geometric mean and variation, coordinate conversion and a function plotting algorithm. These are just some of the many programs included. For only $7.50 you can buy the kind of programs previously available only as part of software math package systems for large scale computers. All the programs are written in a restricted BASIC suitable for most microcomputer BASIC packages, and have been tested and debugged by the authors. $8.50.

BITS T.M.
The Microcomputer Bookstore
25 Route 101 West
Peterborough, NH 03458

DIAL YOUR BANK CARD ORDERS ON OUR TOLL-FREE HOT LINE: 1-800-258-5477.

Name ______
Address ______
City ______ State ______ Zip ______
Signature ______

☐ My check enclosed:
Check #: ______
Amount: ______
master charge
☐ My card #: ______
VISA
Expires: ______

	Total
Number of books ______	______
Postage $.75/book or $1.00 outside U.S.	______
Grand Total	______

Prices subject to change without notice. *Dealer Inquiries Invited.* *You may photocopy this page.*

Circle 34 on inquiry card.

bearing paper, moved one step after inscription of each letter. Knaus described his machines in a 1780 book, *Selbstschreibene Maschine.* His machine number 4 was shown at the Paris Exposition of 1937. It now resides in the Vienna Technical Museum.

The Automata of Jacquet-Droz and Leschot

How can one describe machines so marvelously devised and "tutored" (ie: programmed) in their tasks that they rival the actions of human beings proficient in the art the machine imitates? One can compare them to humans and the analogy is intriguing, but humans are born with the necessity to learn many advanced action patterns and the automata were able to perform several advanced action patterns directly after construction. And humans age and die while the machines are two centuries old and act as well as the day they were set in place. They are seemingly flawless, ageless, potent and wise. And if you compare them to spirits you will be very nearly right, for they are shaped to resemble otherworldly creatures: cherubs or angels. If the compactness, beauty and simplicity of their mechanism with its nearly perfect functioning leads you to compare them to fine watches, you will be very nearly right again, for their builders were first of all horologists. They were the family of Jacquet-Droz (two brothers and a son) and Leschot, their master mechanician.

Long involved in making elaborate timepieces in Geneva, Jacquet-Droz the younger may well have been influenced by word of Knaus' writing automaton. The Writer, Draftsman and Musician he designed and constructed, were placed on display simultaneously in 1774, and they have charmed every person who has seen them. They are on display in the Museum of Automata, in Neuchatel, 30 miles east of Geneva in western Switzerland. Consider the fact: here are devices seen and admired today, as well as by the courts of Louis XV, Louis XVI, George III, Napoleon and even by Franklin and Jefferson.

The Writer writes a preset text of 40 letters and spaces in about the same time and with quite a bit more skill than it might be written by an 8 year old child. The Draftsman draws a series of stored pictures, any one you choose, about as well as a gifted child of 12 years might do, while the Musician plays five melodies on her harmonium, as a musical child of 10 years might do. They have been performing these feats for 204 years.

Two Bytes Are Better Than One

TMS 9900
16BIT
MICROCOMPUTER
SS-16

SUPER STARTER SERIES

FLOPPY DISK DRIVES

COLOR VIDEO BOARD

4800 BAUD DIGITAL CASSETTE

TECHNICO INCORPORATED

THE FULL POWER OF THE **16-BIT TMS 9900 MICROPROCESSOR** IS NOW AVAILABLE WITH THE UNIQUE COMBINATION OF RELIABLE HARDWARD AND FAST, EASY TO USE SOFTWARE IN THE **TECHNICO SS-16.** WITH MINICOMPUTER PERFORMANCE THE **TECHNICO 16-BIT MICROCOMPUTERS** ARE AVAILABLE FROM THE **SINGLE BOARD SUPER STARTER SYSTEM AT UNDER $400** TO THE FULL **SS-16 WITH UP TO 65K BYTES OF MEMORY, MINI-FLOPPY OR FULL FLOPPY DISKS, A 4800 BAUD DIGITAL CASSETTE, 64 COLOR VIDEO BOARD OPTION, RS232 AND 20 MA CURRENT LOOP** ALL COMBINED WITH ONE OF THE INDUSTRY'S FASTEST BASICS AND A FULL ASSEMBLER, EDITOR, LINKING LOADER PACKAGE. SYSTEMS ARE AVAILABLE COMPLETELY ASSEMBLED AND TESTED OR IN UNASSEMBLED **TEC-KIT™** FORM. EXPLICIT MANUAL INCLUDED OR AVAILABLE SEPARATELY AT $35. TO LEARN MORE...JUST TEAR OFF A PIECE OF THIS AD AND RETURN TO TECHNICO OR CALL OUR HOTLINE 1-800/638-2893 OR YOUR LOCAL DEALER.

EUROPEAN MODELS AVAILABLE THROUGH TECHNICO INTERNATIONAL



9130 RED BRANCH RD.
COLUMBIA, MD. 21045
PHONE 301-596-4100

DOMESTIC SALES



2442 N. LEXINGTON ST.
ARLINGTON, VA. 22207
PHONE 703-538-4000
TELEX: 64100 SOLIDSTA

SALES OUTSIDE CONTINENTAL U.S.

Circle 372 on inquiry card.



Photo 1: Four successive es drawn by the Maillardet machine, showing the consistency of the mechanical drive system as well as the variations in letter width that were possible using a quill pen.

Photo 2: Enlargement of letter y drawn by the Maillardet writing and drawing automaton. Each subdivision of the grid is 1 mm.

The Writer

The Writer is 28 inches (71 cm) tall. Carved of wood and painted, this automaton produces "an unusual impression of life" similar to top quality wax figures. He is clothed in a flowing robe and is seated on a Louis XV stool at a mahogany desk. His right hand, poised an inch above the desk and writing tablet, holds a short tube in which a quill pen is fixed. When the mechanism is activated the Writer raises his hand, swings it laterally, dips his pen into the inkwell fixed to the right margin of the desk, shakes the hand twice to clear the pen of excess ink and pauses. Another touch on the mechanism and he begins to write, forming letters with slow, patient care.

After each letter, the pad of paper moves to the left by an amount sufficient to leave space for the next letter, but more for a wide letter or a capital than for is and ls and fs. He can write 40 different letters on two or three lines, and there is programming for several pen dips. Most remarkable is the provision for the unit to vary the pressure of the pen so that the letters produced are weighted, formed of thick and thin strokes.

Except for the few levers controlling movements of the paper tablet, all of the automaton's mechanism is contained in the torso, accessible from the back. There are two parts of the mechanism, and they interact with each other. The first is a cluster of letterforming cams on a common shaft, the cam follower of which rides on a carriage that slides on rails so it can cover the length of the cluster to settle on the rim of the desired cam. There are actually three cam followers and three cams provided for each letter. Two govern movements of the right arm and the third regulates pen pressure for varying the stroke width.

The second portion of the mechanism is the text selector, a disk 4 inches (10 cm) in diameter at the bottom of the cam cluster shaft. The rim of the text selector disk is divided into 40 sectors, or an angular wedge of 9° per sector. The sectors are not fixed, but rather slide radially when one of their 40 screws is turned. In this way the radius of the disk can be varied sector by sector, giving the appearance of a snaggle toothed gear. Each sector in turn regulates the position of the cam follower carriage (with its three cam followers) according to where that sector is set. Thus the text selector disk selects which set of three cams will be employed, and the letter those three cams control is the letter the right arm inscribes. Changing the text is as easy as turning 40 screws to just the right position. The zero radius (baseline) position of the text disk appears to control the pen dipping mechanism, so you can set up as many pen dips as you wish at the loss of a letter or space for each one.

Control is handed back and forth between the text selector disk and the letter forming cam cluster. Either one or the other operates at a given moment, but the text disk is stationary almost all the time (moving in jumps) whereas the cam cluster that forms the letters is moving most of the time (halting only to permit the text selector to turn to its next position and choose the next letter). An intriguing point, for 1774, is that the surfaces of greatest wear (the three cam follower bearing points) are apparently jewelled with ruby so that the high pressures (probably a 40:1 lever ratio, or more) will cause minimal wear and distortion of the letter shapes over time. All this machinery is said to be quite sensitive to temperature changes.

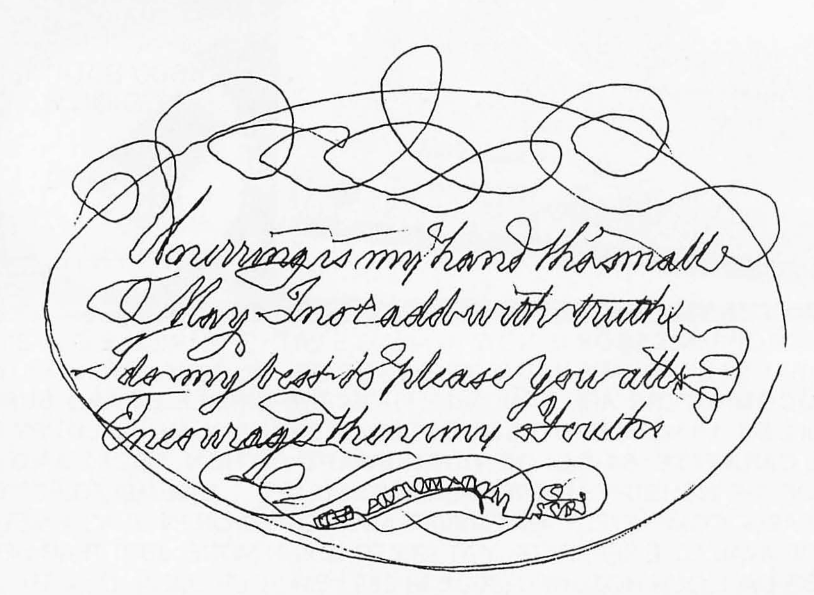


Figure 2: An example of the writing created by Maillardet's writing and drawing automaton using a ballpoint pen. "Unerring is my hand tho' small/ May I not add with truth/I do my best to please you all/Encourage then my Youth." Courtesy Franklin Institute. (For a more complete discussion of this automaton, see *"Philadelphia's 179 Year Old Android" on page 90.)*

The Microcomputer Bookstore



25 Route 101 West, Peterborough NH 03458

Best Sellers

* —**Getting Involved With Your Own Computer** by Leslie Solomon and Stanley Veit answers the questions: "What can small computers do? Which is best for my purposes?" Whether your interest is business applications, word processing, education, security, etc., this lucid text will bring you in touch with an exciting new world destined to affect us all. $5.95.

* —**The Thinking Computer: Mind Inside Matter** by Bertram Raphael. Artificial intelligence, or AI, is the branch of computer science concerned with making computers "smarter." It is a growing, vital field that is, unfortunately, the subject of much popular misunderstanding. This book is a lucid introduction to AI that does much to overcome this misunderstanding. With a minimum of technical jargon, it discusses the capabilities of modern digital computers and how they are being used in contemporary AI research. It discusses the progress of AI, the goals, and the variety of current approaches to making the computer more intelligent. $6.95.

* —**The First Book of KIM** edited by Jim Butterfield, Stan Ockers, and Eric Rehnke. Attention KIM users! Here is the book you've been waiting for. In it you'll find a beginner's guide to the MOS Technology KIM-1 microcomputer as well as an assortment of games including Card Dealer, Chess Clock, Horse Race, Lunar Lander and Music Box. Also featured are diagnostic and utility programs for testing both the computer and external equipment (such as cassette recorders), and chapters on expanding memory and controlling analog devices. 176 pp. $8.95.

* —**Periodical Guide for Computerists,** January-December 1977, by E. Berg Publications. This is a comprehensive index of all the articles, book reviews, editorials, letters, record reviews, and miscellaneous small inserts and notes from the top 25 magazines in the field. Several thousand articles are grouped into over 60 subject categories that are listed alphabetically for easy reference. At the back is an author index, including the major areas of their expertise. An indispensable guide for anyone in the fields of personal computing, amateur radio, and electronics. 72 pp. $5.00.

* —**Scientific and Engineering Problem Solving With The Computer** by William Ralph Bennett Jr is one of the most exciting books we've seen in years. Besides teaching BASIC, this lively, lucid book presents a wealth of imaginative and unusual applications programs taken from many disciplines (A sample exercise: "Using the algorithm in the text with the pair-correlation matrix from Hamlet, compute the most probable diagram path which starts with the letter T"). The exercises run the gamut from random processes to the dynamics of motion, from entropy in language to the Watergate problem. You'll discover BASIC applications in lasers and in the Fourier series and the law (!). In its diversity and elegant style, it ranks with Donald Knuth's works as a milestone in the art of computing. Hardcover $19.95.

* —**Computer Power and Human Reason** by Joseph Weizenbaum. This book is one which should be purchased or read for several reasons. If you're presently a programmer by trade or skill, you'll see a philosophy of computer use and abuse propounded. It's genuinely interesting, and definitely provocative if you reference the storm of letters, counter letters and counter counter letters which this book produced in the Association for Computing Machinery's *SIGART* newsletters during 1976. If you're a novice to the field, the tutorial and explanatory chapters of this book, which are aimed at the layperson, will serve as an excellent background source which is also eminently readable. This includes an excellent and low level explanation of what an algorithm is, and how computers go about executing effective algorithms. $5.95.

Praised by many critics as the best books in their field, **The Art of Computer Programming, Volumes I, II and III,** are part of a projected seven volume omnibus survey of computer science now being completed by Donald E Knuth.

* —Volume I, **Fundamental Algorithms,** begins with a thorough discussion of the mathematics used in computer programming, followed by a treatment of information structures, stacks, arrays, linked lists, dynamic storage allocation, and trees. 634 pp. $21.95.

* —Volume II, **Seminumerical Algorithms,** is concerned with random numbers, statistical tests, random sequences, as well as arithmetic (floating point and multiple precision), polynomials, and rational arithmetic. 624 pp. $21.95.

* —Volume III deals with **Searching and Sorting,** and as the name implies, the emphasis is on algorithms for sorting, including combinatorial properties of permutations, internal sorting, optimum sorting, and external sorting. Also included is a section on sequential searching, hashing, digital searching, and more. 722 pp. $21.95.

A hypothetical assembly language called MIX was developed by the author to illustrate programming examples throughout the series. MIX is easily convertible to other assembly languages.

Professor Knuth writes with style and wit. This classic work belongs on the reference shelf of everyone seriously interested in computer science.

For your convenience in ordering, please use this page plus the order form on page 99. You may photocopy this page.

DIAL YOUR CHARGE CARD ORDERS TOLL-FREE 1-800-258-5477.

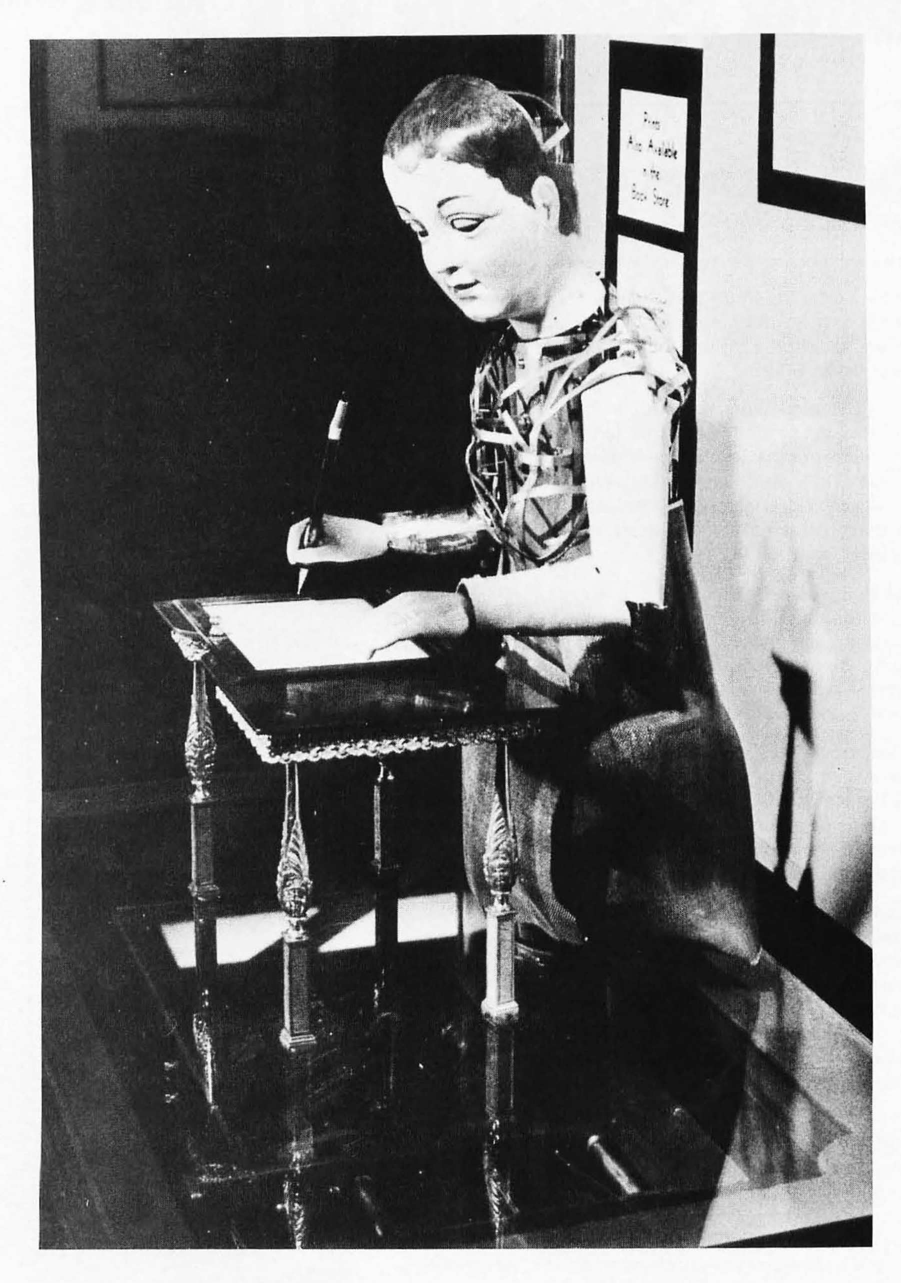


Photo 3: The Maillardet writing and drawing automaton sans clothes. The body of the figure is made from strips of brass and wire; the glass floor reveals the mechanism. A modern pen is shown in the automaton's hand. Courtesy Franklin Institute.

The machine is about 30 inches high and represents a child kneeling before a desk with pen in hand. It is capable of transcribing poetry and drawing intricate figures.

A point which is obscure to me is that the letter forming cams are alleged to operate on a polar coordinate system. Suppose the letters are formed on X-Y coordinates. Photo 2 is a greatly magnified letter superimposed on a grid of 1 mm lines. Now you can appreciate the delicacy of the mechanism, for it is clear that a deviation of ±0.25 mm at any point will make a very different looking letter. (Incidentally, at a 40:1 lever ratio, a 0.25 mm movement at the pen is equivalent to 0.00625 mm on the cam face.) Clearly, the letters as inscribed on paper are well within this deviation (see photo 1 and figure 2). Look how the es from several different words are exact duplicates: Probably the deviation is within about a tenth of that figure (ie: ±0.025 mm).

The mechanism is analog, of course, but if it were digitalized, the scale applied (resolution) has got to be less than 0.025 mm per bit, or in a letter of 8 mm height and 4.5 mm width:

$$\frac{8}{0.025} = 320 \text{ bits for height}$$

$$\frac{4.5}{0.025} = 180 \text{ bits width}$$

A grid of 320 by 180 equals 57,600 points, which would be the upper margin of the error. The limit is plus and minus this, so each letter may be digitalized with 57,600/(2x2) = 14,400 points. But that is the amount for each letter, and we have 26 of them, which is 14,400 x 26 = 374,400. Adding upper case letters, the proper figure is 14,400 x 52 = 748,800 bits to digitalize the entire alphabet within the limits of error the machine consistently displays. You may wish to adjust the figures slightly because not all letters are the size of the y, and hence do not require as much storage of information (see photo 2). However, many letters fall below the line, and the capitals are larger than all the lower case, so it evens out. We have not taken account of the stroke shaping bits, which might require 4 to 6 more increments of information. Altogether, the machine's "read only memory" has over three quarters of a million 1 bit bytes stored within it!

The Draftsman was constructed to resemble the Writer, and works in practically the same manner except that the tablet of paper is fixed, and the arm holds a pencil instead of a pen. The device moves under guidance of a cam cluster and draws designs in segments with pauses while the mechanism shifts from one cam pair to the next. During these pauses the Draftsman blows a puff of air from his lips to disperse the graphite debris. I would estimate that there might be 20 or more cam pairs for each of the four designs (there are no depth cams) on a slip of paper about 2 by 3 inches (5 by 7.5 cm). The designs were simplified reproductions of popular etchings of the age: cupids in chariots being hauled by butterflies, etc; and the head of Louis XV. The little Draftsman appears to have elicited a good deal more excitement than the Writer, but he was actually easier to construct, since the builders profited from their earlier experience with the Writer and simplified the mechanism.

Assume that the Draftsman's paper is 50 by 75 mm, that any point on it could play a part in the design, and that it was necessary to provide a mechanism that could discriminate between lines as close together as 0.5 mm (ie: to a tolerance of ± 0.25 mm). You end up with a grid of 50/0.25 by 75/0.25 = 200 x 300 = 60,000 points that may be encoded. These were parcelled out among 20 "read only memory" cams. The total information contained in the machine would be 60 K bits by 4 designs = 240 K bits. The total information storage was much less because the eye can accept more line deviation in a drawing than in the formation of a letter.

The Musician is the triumph of automata that counterfeit life. She is 42 inches (1.07 m) high, seated at her instrument with a pleasant expression on her face. Her clothing is rich satin brocade in the elaborate style of the period, and her coiffure is impressive. She consecutively plays five pieces on her instrument, a curious device rather like a harmonium but called by some accounts a flute-organ, suggesting tuned pipes instead of metal reeds. The keyboard consists of two arcs of keys, 12 on a side. It is double arc shaped because the musician's arms pivot at the elbows (concealed by lace sleeves on her gown) enabling her to cover all 12 keys with five fingers. The music, or most of it, was composed by Jacquet-Droz the younger, a musician who studied composition with Marchal.

She actually fingers the keys that produce the music! The mechanism to accomplish this feat consists of a connection for each digit, and some extremely clever devices must be employed to enable the arms to swivel while maintaining continuity for the digit controlling mechanism. I leave you to contemplate the delicacy of the arrangements of mechanism that trigger each finger in the tiny hands, but keep in mind that this machine is a workhorse; this musician has been playing music for 200 years.

Her programmed movements are startlingly lifelike in the accounts. All the Jacquet-Droz and Leschot automata turn their heads and move their eyes, but this automaton also raises her head to look at the audience, drops her gaze, takes a deep breath, and starts to play. She turns her head as she plays and, swaying from side to side as artists will do, breathes all the while. At the end of a piece she looks up and seems to smile, then shyly lowers her gaze, drops her head, and curtsies.

Other Musicians

In 1784 Maillardet, who was in business with Jacquet-Droz (fils) in London, introduced a new and improved version of a lady musician. She played a sort of piano, perhaps actually a harpsichord, and it is known that she had 17 or 18 melodies in her programming. She was lost in 1833 when sent to St Petersburg together with other automata.

The Dulcimer Player of Roentgen and Kintzing first appeared in 1780, and was said by the magician Robert-Houdin (who repaired her in 1866) to have been designed to resemble Marie Antoinette and emulate her skill with the string dulcimer. This figure is famous for her beauty, and much praise has been lavished on her musical skill, for the instrument is clearly a difficult one to play (and is hardly known in this country). The mechanism is a cluster of cams mounted below the figure, concealed by her gown.

J N Maelzel, mechanician to the Austrian court and later the proprietor of the Chess Automaton, personally designed and had built a life-size Automaton Trumpeter, which he exhibited beginning in 1808. It was destroyed in a fire, about 30 years later. At least two other trumpeters have existed. What remarkable mechanism they must have contained, especially in view of

the praise their performances evoked. None survives.

Maelzel invented and displayed, beginning in 1804, the Panharmonicon, a compound musical mechanism which produced the sounds of flutes, trumpets, drums, cymbals and triangle, and plucked strings, a menage then called Turkish music and much favored by the public. This machine was followed by his Orchestrion, imitating the sound of the military band (which had become popular during the French Revolution). An improved Panharmonicon, with clarinets, violins, and violas added, was so well received that Maelzel commissioned music from Dussek, Pleyel, Weigl, and even Beethoven, whose "Wellington's Victory," opus 91, employing the Automaton Trumpeter as well as the orchestra, had its premiere on December 8 1813, in Vienna. These devices were the first of the programmable multiple instrument machines so popular 75 years later.

A Combination Automaton by Maillardet

It was known that Maillardet, constant collaborator with the Jacquet-Droz and Leschot organization, had constructed a writing and drawing automaton about 1811, which was exhibited in London in 1815, and was owned by several persons until 1833 when it was sent to St Petersburg where it disappeared.

Long ago a resident of Philadelphia mentioned to a staff member of the Franklin Institute that his family owned an automaton that drew pictures and wrote poems. He supposed it to be Maelzel's work. When the owner's house was destroyed by fire, reducing the automaton to a "mass of cams and wheels," the museum acquired it, but it took immense patience and care on the part of the museum restorer, Charles Roberts, to make the machine completely whole. In the restoration process the sex of the automaton was changed. When the time came to sample the machine's program, it was found to be Maillardet's missing automaton (see photo 3 in this article and Charles Penniman's article, "Philadelphia's 172 Year Old Android" in this issue, page 90).

The machine is about 30 inches (76 cm) high, and represents a child (originally a little boy, as alluded to in one verse, and in an 1812 encyclopedia article) kneeling before a desk and holding, since restoration, not a brush but a pen. The mechanism is in the base and consists of a common shaft holding about 60 cams, each one 6 inches (15 cm) in diameter. The whole is driven by a pair of powerful spring motors. Three triplets of cams are devoted to each of the seven productions of the automaton, except that the depth cams are minimally employed. The follower arms, one for each dimension of the drawing, are jewelled and move from pair to pair of cams in the course of one machine cycle (one drawing). The automaton executes its seven productions rapidly, completing one in 7 to 8 minutes. This would appear to explain Maillardet's need to skeletonize the 60 programming cams: they turn rather swiftly (about 3 mm of linear motion per second) and at changeover they must be brought quickly to a halt, then accelerated to working speed again. Storing all information on three pairs of large cams per production would have made grinding the cam faces much easier, and would have minimized the effects of wear compared to a small cam. Shifting to a new program is done by simply sliding the common shaft laterally to set up a new triplet of cams.

Maillardet evidently took it as his task to produce a machine that worked on its productions rapidly and casually, perhaps in the manner of a person inspired. The sketches are marked more by fluency of line than by precision, but they are very sophisticated, as a glance at the ship sketch will show (see page 91). The poetry is interesting and is done more in the manner of a design with scriptwriting than writing in script (see figure 2).

In terms of brute force memory storage, if each of the points 1 mm apart on an 89 by 120 mm paper is to be stored, 10,680 points would be required. But discriminating between points with an error of no more than 1 mm requires ± 0.5 mm precision, resulting in 42,720 points that must be stored on the three triplets of cams. But this is the amount of point storage required for one production. There are seven of them, so the total storage capacity within the machine is 42,720 x 7 = 299,040 points (with ±0.5 mm precision). This figure, the digital equivalent of the analog storage, begins to make the impressive forest of cams seem more useful.

All of the above speaks about the *information capacity* (in terms of a grid of points) necessary to encode the designs and script that our automata can produce by analog means. The great majority of those digital data would not be employed in a display, just as an automaton will not inscribe marks on, say, more than 2 percent of the area of paper available to it. There is a lot of wasted (unused) space in any charac-

Note:

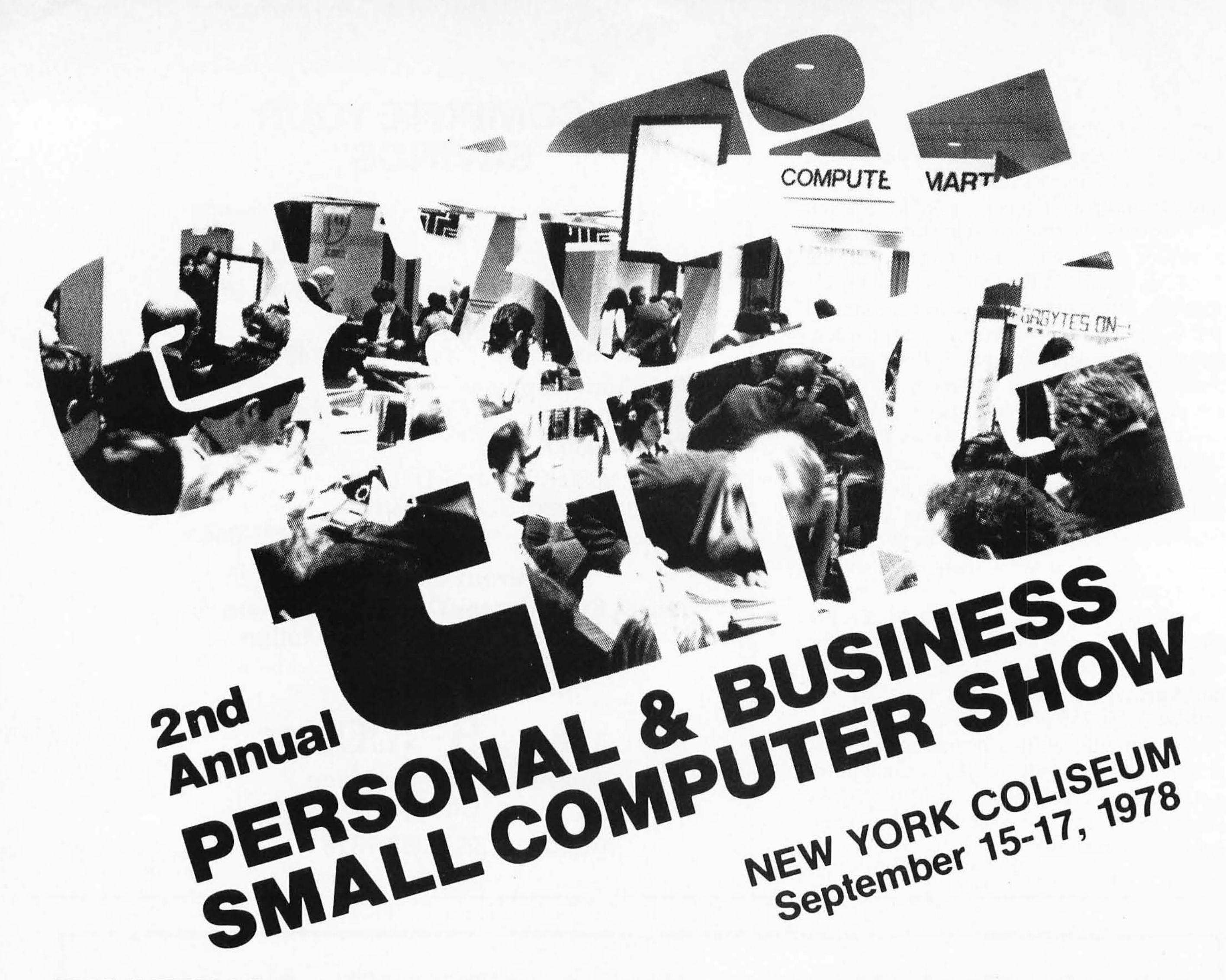
A complete bibliography for this part of "Antique Mechanical Computers" will appear with "Part 3: Human and Machine Action and the Torres Chess Automaton" in September 1978 BYTE.

ter generator. For example, most of the billions of micrograins of silver halide in a sheet of emulsion are not actually developed and play no part in a photographic negative.

The same is true of standard character generator read only memories where the 5 by 7 matrix with its 35 points is the absolute minimum matrix you can employ and still produce recognizable, if not particularly legible, alphanumerics. Even so, 50 percent of these bare minimum 35 points are not utilized for any given character, hence there is 50 percent waste. Premium character generator read only memories are set up to use a great many more points, and their displays are still manifestly coarse in structure ("crude" would not be too strong a word, when you know there is something much better).

Here we are simply making visible the difference between analog and digital modes of storing information. The analog mode is obviously more economical, for there are nowhere near 750,000 jiggles in 20 cams.

Next month, we'll conclude with a final example in this series about antique automata, the chess playing robot of Torres (circa 1911) and some philosophical comments on automata.■



EXCITING

That's the word that sums up the dynamic small computer show in New York that was established last year at the Coliseum. The lecture areas were filled by interested people, the exhibits were great, the attendance was the highest of any small computer show in the country.

The big point is that everyone who came went away happy. Accountants, hobbyists, lawyers, doctors, brokers, retailers, business people, programmers, research scientists and just plain homeowners.

The second big point is that we're doing it again. This time, bigger and better. Top seminars, top exhibits. Small computers from $500 up, software and kits.

Save time. Get your tickets in advance by mail.

NAME ______________________
(please print)
ADDRESS ______________________
City __________ State __________ Zip __________

Send me tickets in advance for Personal & Business Small Computer Show, Sept. 15-17, 1978, in the New York Coliseum.

(Check One) One Day ($5)__ Two Days ($9)__ Three Days ($13)__

Send check or money order (U.S dollars) payable to Personal & Business Small Computer Show to 78 E. 56th St., New York, N.Y. 10022

PERSONAL & BUSINESS SMALL COMPUTER SHOW
78 East 56th Street, New York, N.Y. 10022

Circle 267 on inquiry card.

IN NEW YORK! FREE SEMINARS

Personal & Business Small Computer Show offers the following:

INTRODUCTORY

Introduction to Small Business Systems
Interfacing to the Real World
Introduction to BASIC
Hardware/Software/Systems Tutorial

SOFTWARE

Data Bases & File Handling
Introduction to CP/M
Introduction to PASCAL
Introduction to APL
SBS: A New Data Base Manager, Language and Operating System—a Software Breakthrough that Gives Minicomputer Power to a Micro—for Business Applications, from Computer Mart of New Jersey

SMALL BUSINESS APPLICATIONS

Text Editing/Word Processing
General Ledger/Acc'nts Receivable & Payable
Mailing Lists

RECREATIONAL APPLICATIONS

Robotics
Music
Computer Games
Art, Graphics & Animation

HOUSEHOLD APPLICATIONS

Budget Management
Investment Analysis

PROFESSIONAL APPLICATIONS

Patient & Client Billing

EDUCATIONAL APPLICATIONS

Light Pen Graphics
Simulations & Gaming

All seminars run about 50 mins. and are scheduled twice. Write for daily and hourly schedules, special hotel rates, etc. Lectures subject to change.

IN NEW YORK! AVAILABLE TUTORIALS

IEEE COMPUTER SOCIETY

Tutorials are offered by the IEEE Computer Society in conjunction with the show. Each tutorial costs as follows: members, $45; non-members, $50, and students, $35.

For further information on the technical background required, the content of the tutorials, or the IEEE, write to Dr. Daniel R. McGlynn, 71 N. Moger Ave., Mt. Kisco, N.Y. 10549

5 TUTORIALS BY SYBEX

Microcomputer Selection—Sept. 16
Intro to Microprocessors—Sept. 15
Programming Microprocessors—Sept. 16
Designing a Micro System—Sept. 16
*Interfacing Techniques—Sept. 17

***Prerequisite—Intro to Microprocessors**

All Sybex tutorials run for three hours. Cost is $29.95 each up to Sept. 8. $35 each at the door. All are given in the Coliseum. Show ticket is needed for entry. Please use coupon for advance ticket.

(For registration and more details write to Sybex, 2020 Milvia, Berkeley, CA 94704) TEL: (415) 848-8233

GENERAL ATTENDANCE INFORMATION

The show will occupy the fourth floor of the New York Coliseum located at 59th St. & Columbus Circle. There is a parking garage in the building and the site is served by many subway lines and buses.

Visitors from out of town may obtain special hotel rates from our headquarters hotel, the Barbizon Plaza, just two blocks from the show. Singles $44, doubles $52. Identify yourself as attending the show and request those rates.

Admission to the show at the door is $5.00 per day, per person. Discount tickets for groups of ten or more available in advance only by mail at $4 per person per day. Enclose check or money order for all ticket orders payable to Personal & Business Small Computer Show. For further information call (212) 753-4920.



PERSONAL & BUSINESS SMALL COMPUTER SHOW

78 EAST 56TH STREET
NEW YORK, N.Y. 10022

New York Coliseum
Sept. 15–17, 1978

SHOW HOURS

Sept. 15—noon to 8 p.m.
Sept. 16—10 a.m. to 8 p.m.
Sept. 17—noon to 7 p.m.

Languages Forum

In Praise of PASCAL

David A Mundie
104B Oakhurst Cir
Charlottesville VA 22903

As has been pointed out in these pages before, personal computing will never achieve its full potential as long as our state of the art machines are hobbled down with a language as far from state of the art as BASIC is. Some have argued for designing a special high level language for microprocessors, but I personally fail to see why we don't just implement PASCAL and be done with it. I would like to look briefly at the language itself and try to explain why it seems the logical choice to me.

I am an applications programmer with no theoretical interest in computing whatsoever. What I like about PASCAL is not the theory of its design, though that seems sound enough, but rather the fact that it lets me formulate my problems in my own terms. In PASCAL more than in any other language I know, I can remain on the abstract, algorithmic level where, as a human being, I function best. Because of this pragmatic bias, much of what follows will be an informal discussion appealing to the reader's intuitions rather than a technical demonstration. I shall use BASIC for comparative purposes, since it is the tyrant in the field.

I find PASCAL easy to use because it allows me to define new data types which express my data meaningfully. It provides control structures with which I can express what I want done to my data clearly and naturally. PASCAL allows and encourages me to formulate my thinking in a structured way. Let us examine these three aspects of PASCAL in reverse order.

Program Structure

PASCAL is a resolutely structured language. A PASCAL program is structured into blocks. Each block bears a heading which gives it a name and specifies its parameters. Roughly speaking, a block consists of a definition part, in which constants, types, variables, and subroutines are defined, and an action part, which contains the algorithm of the block. This rigorous separation of data definition and algorithm expression is partly responsible, it seems to me, for the greater legibility of PASCAL compared to ALGOL.

Subroutines are themselves block structured and may thus be nested within one another. This allows the declaration of "local" variables and subprograms, meaning that storage may be allocated efficiently; yet it is easy to guard against unwanted side effects.

What does all this mean for the practicing programmer? The answer may perhaps best be seen in the light of a claim recently repeated by David Higgins in the October 1977 BYTE ("Structured Program Design," page 146). Higgins presents the now well established arguments in favor of structured programming, but goes on to contend that once a program is designed in a structured way, using for example Warnier-Orr diagrams, "it does not matter what programming language you code it in." This assertion seems pretty improbable on the face of it, and if true it would be a powerful argument against PASCAL. I think that a rapid examination of two test cases will show it to be quite unjustified.

Let us take our test cases from the "bug" program which Higgins uses as his own example. Higgins would have us break the program down into three parts, as expressed in the following Warnier-Orr diagram:

```
              ( begin program
bug program  <  games (1,g)
              ( end program
```

Nothing in the BASIC listing which accompanies the article even remotely suggests this overall algorithm. Look at what we might have in an equivalent PASCAL program:

```
program bug;
   begin
      beginprogram;
      games;
      endprogram
   end.
```

Need I point out that to all intents and purposes the PASCAL program *is* the Warnier-Orr diagram, with only a few notational

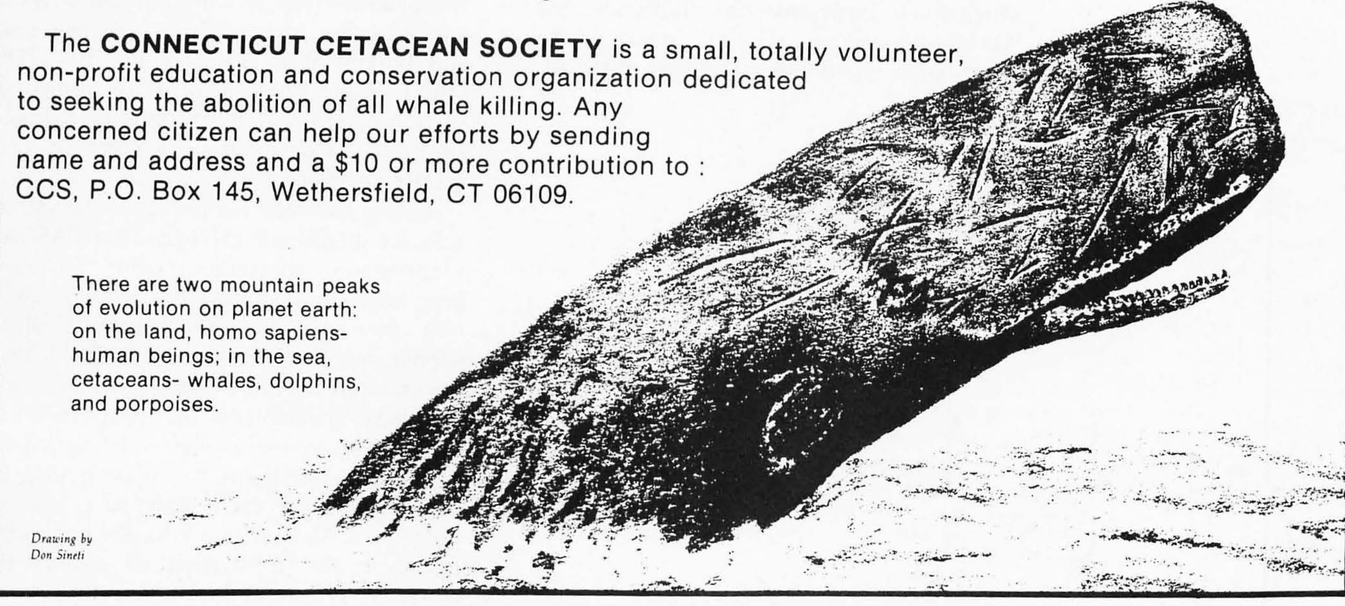
SAVE THE WHALE

The world's best computer may be inside a Sperm Whale's head.

The Sperm Whale has the largest brain of any creature that has ever existed on our planet. The brain of this 18-meter marine mammal weighs up to 9 kilograms. It uses echo-location to find giant squid at ocean depths of over 1,000 meters. More than 13,000 sperm whales are scheduled to be slaughtered this year by agreement of the International Whaling Commission.

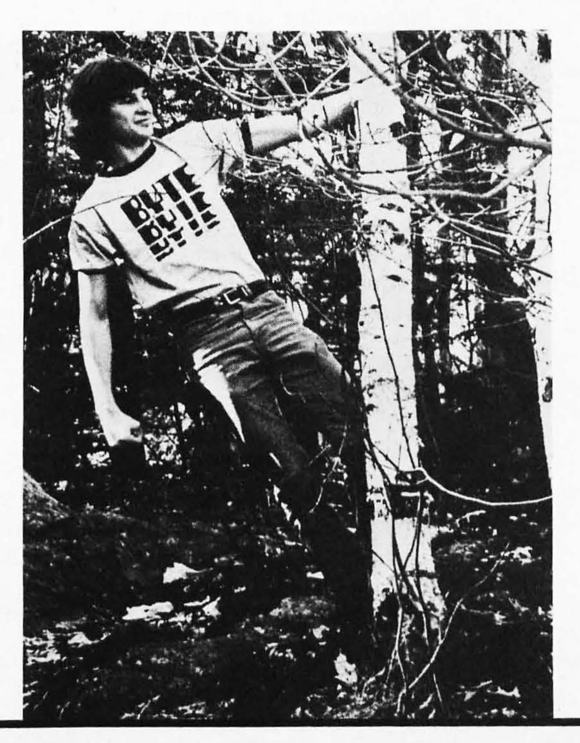
The **CONNECTICUT CETACEAN SOCIETY** is a small, totally volunteer, non-profit education and conservation organization dedicated to seeking the abolition of all whale killing. Any concerned citizen can help our efforts by sending name and address and a $10 or more contribution to : CCS, P.O. Box 145, Wethersfield, CT 06109.

There are two mountain peaks of evolution on planet earth: on the land, homo sapiens- human beings; in the sea, cetaceans- whales, dolphins, and porpoises.



Drawing by Don Sineti

Spring into Season with a BYTE T-shirt



At last! No more wardrobe crises! BYTE T-shirts are here! Now you have the perfect garb for computer club meetings, Altair Conventions, playing Shooting Stars and computer chess. (A pair of trousers from your own closet is suggested as an addition to the BYTE T-shirt. BITS can't do everything for you.)

BYTE T-shirts are of top quality 100% cotton or cotton-polyester. The original design, by artist Judy Lee Rehling, is silk-screened in red on white shirts with blue trim on collars and sleeves, or on blue heather shirts.

BITS, Inc.
70 Main St.
Peterborough NH 03458

Dial charge card orders toll free 1-800-258-5477.

Please send me: ___extra large ___large ___medium ___small

___ blue heather
___ white with blue trim and red letters

T-shirts @ $5.50 each (includes postage and handling).

Total enclosed $________

Bill MasterCharge No.____________ Exp. Date______

Bill BankAmericard No.____________ Exp. Date______

Name__________________

Address__________________

City________ State______ Zip______

Signature__________________

In unusual cases, processing may exceed 30 days.
Prices shown are subject to change without notice.

differences such as the replacement of the brace by the symbols **begin** and **end**? Are we really asked to believe that this one to one correspondence between the problem and the program does nothing to simplify the programming task? On the contrary, it simplifies matters enormously.

Considerations of space prevent me from giving the rival BASIC and PASCAL versions in full. Another striking example is presented in figure 1 and listings 1 and 2, which show the Warnier-Orr diagram for the "turn" subprogram, Higgins' coding of the subprogram in BASIC, and the PASCAL equivalent. Higgins calls his BASIC coding "simple and straightforward." Tastes differ but that is a phrase I would have reserved for the PASCAL version. Higgins has had to fake truly structured programming in a language which fights his efforts every step of the way, and the results are tortured and confusing. In contrast, the PASCAL coding is, once again, a nearly perfect reflection of the Warnier-Orr diagram, so much so, in fact, that most PASCAL users will probably feel, as I do, that the diagrams are a useless intermediary step, less clear and bulkier than the program itself. The intent of the PASCAL program segment is so transparent that in my opinion it could almost be understood by a complete programming novice.

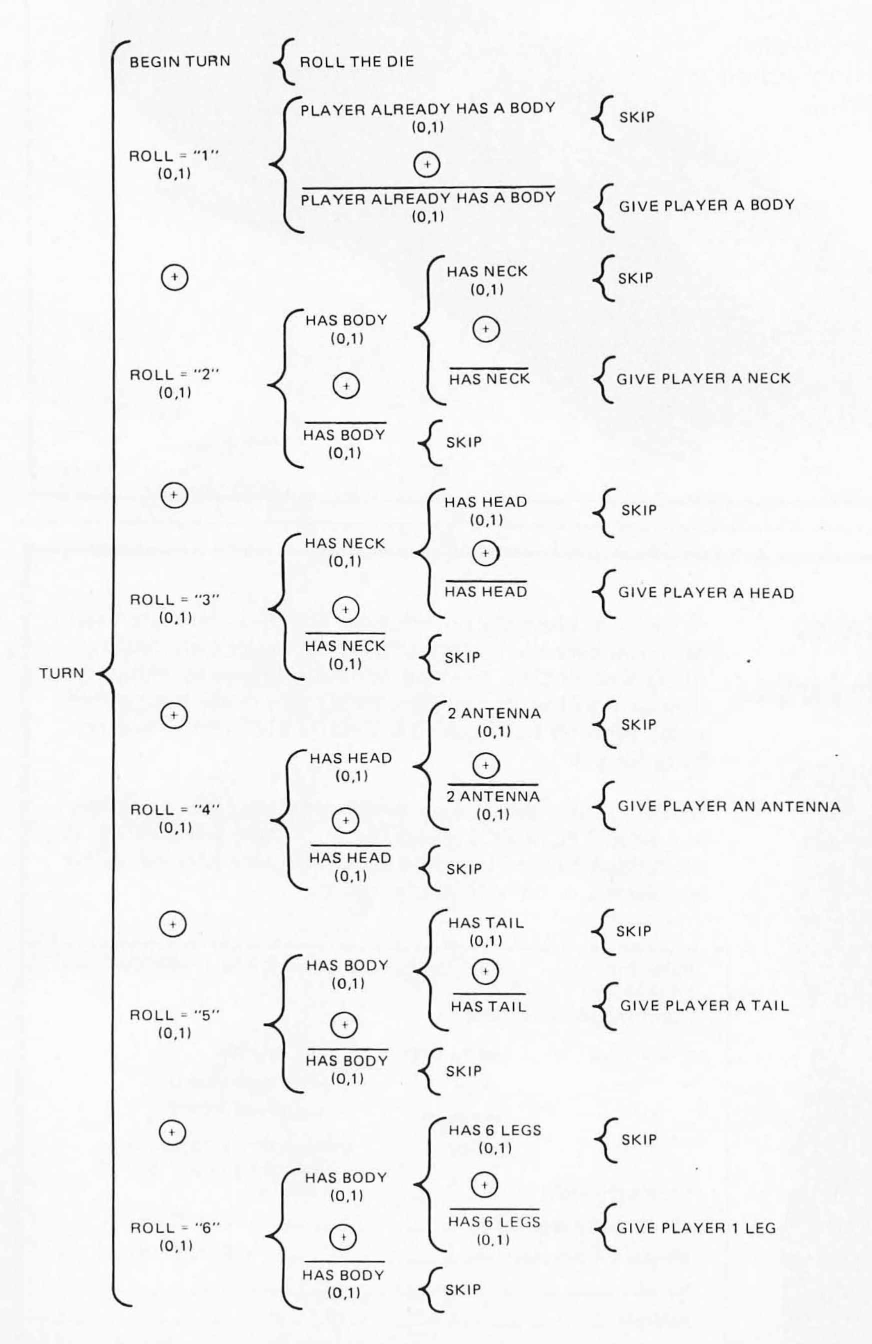


Figure 1: Warnier-Orr diagram for subprogram "turn" of the bug program. This is clear, but note how much bulkier it is than the PASCAL program in listing 2. The Warnier-Orr diagram won't even run on a computer.

Before leaving the topic of program structure, we should perhaps remark that PASCAL subprograms (procedures and functions) bear names, not numbers, virtually eliminating the need for the comments which pepper any well documented BASIC listing. Furthermore, because PASCAL subprograms can have parameters, the programmer is encouraged to use a single subprogram for a single task. Higgins has written separate subprograms for each body part, whereas for a PASCAL user it is virtually impossible to resist the temptation of passing the arrays body, neck, head, etc, to a single procedure "give" as parameters.

Algorithm Expression

Program structure alone does not explain the relative clarity of the PASCAL listing in listing 2. We may also use that listing to illustrate the tools which PASCAL provides for expressing algorithms.

Logical operators: PASCAL provides the logical operators (**and, or,** and **not**) which are so painfully lacking in BASIC and without which expressing an algorithm is so clumsy. The use of the operator **and** in the turn subprogram is a good example; or the reader may want to express "if (x=1) or ((y>2)and(z=3)) then. . ." in BASIC.

Conditional statements: PASCAL's **if** structure groups statements with the conditions for their execution. The **if** statement is of the form:

```
if<expression>
   then<statement_1>
   else<statement_2>
```

The expression is evaluated as being either true or false. If it is true statement_1 is performed; otherwise statement_2 is performed. Suppose the expression is: X=1. In English the **if** statement translates to:

> if X equals 1 then perform statement_1; else perform statement_2.

PASCAL offers a very flexible **case** statement which is remotely related to the computed GOTO statement to be found in some BASICs. It is much more powerful because, among other things, selector values need not be contiguous, and actions are grouped with the conditions for their execution. A good example of the **case** statement's clarity is to be found in the procedure "turn," where the action taken depends on the value of roll.

Repetitive statements: BASIC provides only one repetitive control structure: the FOR statement. But there are innumerable situations where we do not know ahead of time how many times a given action is to be repeated. In such cases BASIC users have two choices. One is to set up a dummy FOR statement with a jump out of it when a certain condition is met: whence the ubiquitous "FOR I=1 TO 9999" statements in BASIC programming. This is bad because it seriously disguises the intention of the algorithm. One's natural expectation is for such a loop to be executed 9999 times, but that is not the case. The other solution is for the programmer to fake an appropriate control structure with GOTOs or conditional jumps. That is what Higgins has done in his program to express the fact that the computer and the human take turns until the game is over:

```
210 REM TURNS (1,T)
220 LET EGAM = 0
230 GOSUB 390
240 IF EGAM = 0 THEN 230
250 REM END GAME
260 GOSUB 1150
```

This is no doubt the best one can do in BASIC, but just consider how much more elegant the PASCAL version is:

```
repeat turns until endofgame
```

This is typical of the way in which PASCAL's control structures make algorithm expression a source of joy rather than a contortionist exercise. In addition to the **repeat** statement, PASCAL offers a **while** statement for the case when an action is to be repeated as long as a condition is true.

Data Definition

Now that we have seen how much easier it is to express what one wants done to data in PASCAL than in BASIC, let us turn to the wonderful data types which PASCAL makes available for manipulation. Data types are the programmer's buffer between his abstract formulation of an algorithm and the messy realm of bit level details where that algorithm will eventually be executed. PASCAL makes defining new types a trivial task. Once a new data type is defined, it is in effect indistinguishable from a predefined type and may be used in any way a predefined type may be. We leave BASIC behind at this point, since that language has no facilities for creating new types.

The bug program was too simple to provide examples of data structuring, so we shall have to turn elsewhere. Being a birdwatcher, I shall replace the traditional "Christmas card list" example by a bird data bank. I can do no more than skim the surface, so I ask the reader's indulgence if some of the listings are not fully explained. I am not trying to teach PASCAL, but merely to spark intuitions.

PASCAL distinguishes between simple and structured types. Let us examine each in turn.

Simple types: These are the basic building blocks of which any structured type, no matter how complex, is ultimately composed. In addition to integer, real, and character types, PASCAL offers two additional simple types which as far as I'm concerned come close to exhausting the simple types needed in a general purpose language. The first is the defined scalar type, and is defined by simply listing the values which a variable of the new type may take on.

```
490 REM TURN SUBROUTINE
500 REM PLAY=1;PLAYERS TURN–PLAY=2;COMPUTERS TURN
510 REM ROLL DIE
520 LET ROLL = FIX@(((RND(0))*6.0))+1
530 PRINT:"ROLL IS A",ROLL
540 IF ROLL = 1 THEN IF BODY(PLAY)≠1 THEN GOSUB 690 ELSE;ELSE;
550 IF ROLL = 1 THEN 650
560 IF ROLL = 2 THEN IF BODY(PLAY) = 1 THEN IF NECK(PLAY)≠1 THEN GOSUB 760
570 IF ROLL=2 THEN 650
580 IF ROLL=3 THEN IF BODY(PLAY)=1 THEN IF NECK(PLAY)=1
    THEN IF HEAD(PLAY)≠1 THEN GOSUB 820
590 IF ROLL=3 THEN 650
600 IF ROLL = 4 THEN IF HEAD(PLAY)=1 THEN IF ANTE(PLAY)≠2
    THEN GOSUB 880
610 IF ROLL=4 THEN 650
620 IF ROLL = 5 THEN IF BODY(PLAY)=1 THEN IF TAIL(PLAY)≠1 THEN GOSUB 940
630 IF ROLL=5 THEN 650
640 IF ROLL = 6 THEN IF BODY(PLAY)=1 THEN IF LEGS(PLAY)≠6 THEN GOSUB 1000
650 LET A=3
660 RETURN
```

Listing 1: BASIC listing for Warnier-Orr diagram in figure 1. This is the best one can do in BASIC, but is still a far cry from the clarity of the PASCAL listing.

```
procedure turn;
 begin roll:=trunc(random(1)*6)+1; writeln('roll is a',roll);
  case roll of
    1: if(body[player] ≠1)then give(body);
    2: if(body[player] =1)and(neck[player] ≠1) then give(neck);
    3: if(neck[player] =1)and(head[player] ≠1) then give(head);
    4: if(head[player] =1)and(ante[player] ≠2) then give(ante);
    5: if(body[player] =1)and(tail[player] ≠1) then give(tail);
    6: if(body[player] =1)and(legs[player] ≠6) then give(legs)
  end
 end;
```

Listing 2: The PASCAL listing equivalent to listing 1. Note the clear affinity between the listing and the Warnier-Orr diagram. Notice that arrays are indexed using square brackets.

Suppose I need a data type for the various habitats in which a bird may appear. In PASCAL I write:

```
type h = (ocean,rivers,fields,suburbs,forests,
          mountains)
```

A variable of type h may take on any of the values listed. This means that while programming I may continue to think in terms of habitats, and am not forced to descend from that abstract level and think in integers, as I would have to do in BASIC. This also makes for virtually self-explanatory programs. Compare "IF HABITAT=3 THEN. . ." with the much more transparent "**if** habitat=fields **then**. . . ."

The second simple data type is the Boolean, and is extremely useful in programming since one is constantly controlling program flow with Boolean expressions. Boolean variables take on the values true and false. Languages without such variables must make do with integers, which muddles things since one's natural expectation is for integers to count something. The PASCAL user may simply write "**if** good **then**. . .", which is the way we think; the BASIC programmer must write "IF GOOD = 1 THEN. . .", which is alien to the way we think.

A large part of PASCAL's elegance comes from the fact that in most contexts these simple or scalar types may be used indifferently. Thus for example the type h as defined above could be used as the index variable in a **for** statement:

```
for habitat := ocean to mountains do
```

or in a case statement, or as the index type of an array:

```
if foundin [fields] then
```

Furthermore, functions may return any scalar type: we have already seen the function "endofgame" which returns a Boolean value.

Structured types: In addition to the simple types, PASCAL offers five different structuring methods: arrays, records, sets, files, and pointers. These different methods may be combined in virtually limitless ways. One may have files of arrays, pointers to records, arrays of sets, pointers to files of arrays of records, and so on. This extreme flexibility of data structuring methods is one of PASCAL's most exciting features. The type array should be familiar, but let us look briefly at the other four structured types.

Sets: Each bird in my hypothetical data bank has associated with it a set of habitats in which the bird may be found. Having defined the type h as above, all I need to do to set up a variable habitats which will be a set of different habitats is to write:

```
var habitats: set of h
```

When constructing the entry for the robin, I will write:

```
habitats := [fields,suburbs]
```

thus assigning to the robin the set of habitats containing the two elements fields and suburbs. When going on a trip to the mountains, I can test whether mountains are in a given bird's set of habitats by the following simple test:

```
if mountains in habitats then
```

Imagine trying to do this in BASIC. PASCAL provides a variety of set operators which allow set manipulation in all its generality.

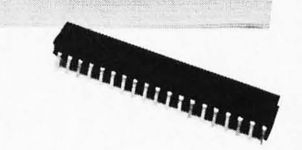
Records: Let us imagine that each entry in my data bank will contain the bird's name, its length, and a set of habitats where it may be found. The entry cannot be an array, since components of arrays must all be of the same type. The appropriate data type is the record, defined in PASCAL as follows:

```
type bird = record
              name: string;
              length: real;
              habitats: set of h
            end
```

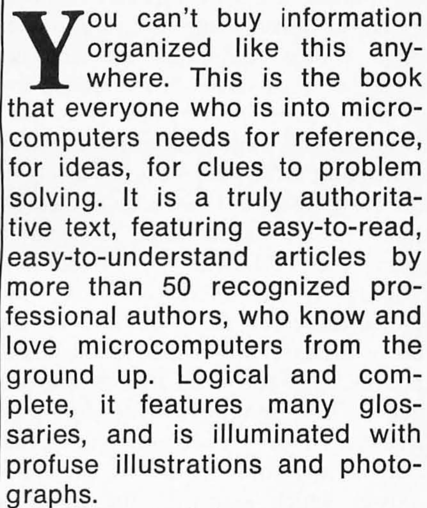
Everything you've ever wanted to know about microcomputers in ONE complete book for only $10.95

Over 400 pages. Full 8½" x 11" size.

THE SCELBI BYTE PRIMER



The ultimate book about microcomputers. Written by experts ... SCELBI and BYTE. Over 400 pages. A collector's item, featuring The Basics from the first 16 issues of BYTE and SCELBI's classic library of books. Your microcomputer bookshelf is incomplete without this priceless edition.



You can't buy information organized like this anywhere. This is the book that everyone who is into microcomputers needs for reference, for ideas, for clues to problem solving. It is a truly authoritative text, featuring easy-to-read, easy-to-understand articles by more than 50 recognized professional authors, who know and love microcomputers from the ground up. Logical and complete, it features many glossaries, and is illuminated with profuse illustrations and photographs.

The Scelbi/BYTE Primer is divided into four logical sections, that take you from point "0" through building and programming your own computer... step-by-step-by-step.

What can you do with a microcomputer? Checkbook balancing. Recipe converting and food inventory. Heating and air conditioning control. Home and business security and management. Playing the ponies. Analysis of the stock market. Maintaining massive data banks. Self-instruction. Toys and games. Small business accounting and inventory. And lots, lots more.

How does a microcomputer do it? Lots of "how to" theory. Introducing you to microcomputer operation. 6800, 6502, Z80 CPU chip capabilities. RAM and ROM memories. Addressing methods.

THE SCELBI/BYTE PRIMER

Over 400 pages. Selected articles from BYTE and SCELBI books. Profusely illustrated. Many photographs. $10.95, plus $1 shipping and handling.

How to control peripherals. Transmission of information to and from computers. Magnetic recording devices for bulk storage. Analog to digital conversion. How a computer can talk. Other I/O techniques. And more.

Order your copy today!

SCELBI COMPUTER CONSULTING INC.	BITS
Post Office Box 133 PP STN	70 Main Street
Dept. B	Peterborough, NH
Milford, CT 06460	03458
	1-800-258-5477

Prices shown for North American customers. Master Charge. Postal and bank Money Orders preferred. Personal checks delay shipping up to 4 weeks. Pricing, specifications, availability subject to change without notice.

All about building a microcomputer system. Over 12 complete construction articles. Flip-flops. LED devices. Recycling used ICs. Modular construction. Making your own p.c. boards. Prototype board construction. Make your own logic probes. Construction plans for 6800 and Z80 computers. Building plans for I/Os — TV and CRT displays, cassette interfaces, etc. Mathematics functions. ROM programmer. Plus much, much more.

How to program a microcomputer. Programming for the beginner. Assembling programs by hand. Monitoring programs. Number conversions. Game of Hexpawn. Design your own assembler. Lots more.

And that's only the beginning! Others have spent millions acquiring the type of microcomputer information found within the 400 pages of **The Scelbi/BYTE Primer.** But, it costs you only $10.95, plus $1 for postage and handling, complete! You know the quality of Scelbi and BYTE. This is your assurance of excellence throughout this MUST text. Order your copy today! And, get one for a friend!

Circle 322 on inquiry card.

This is a simple and logical way of grouping data of different types into a meaningful whole. Given variables robin and redbreast of type bird, a simple assignment statement will set one equal to the other:

```
robin := redbreast
```

To test whether a robin is more than 20 cm long, we would have:

```
if robin.length>20 then
```

and so on. These are simple examples, but they suffice to illustrate the flexibility of the record type.

Files: Now let us suppose that I have 600 entries of type bird in my data bank, and want to make a list of all the birds whose length is greater than 20 cm. It is pointless and wasteful to keep all 600 records in memory for such a task; all I really need is to store them in mass storage and read them in one at a time. In PASCAL what I do is declare a file of records as follows:

```
var fb: file of bird
```

Now, supposing the file to have been written, all I need to perform the task is:

```
reset(fb) ;
repeat if fb↑.length>20
    then writeln(fb↑.name) ; get(fb)
until eof(fb)
```

Reset positions the file at its beginning; get advances it one record; fb↑ is the buffer variable containing the current record; and the writeln statement prints the bird's name. The Boolean function eof tests for the end of the file.

Pointers: Finally, let us suppose that I wish to update the data bank by deleting a bird. It is of course possible to do this by storing all the records in an array, but this is clumsy and inefficient, since all the records following the deleted record would have to be shifted one position. List processing provides a much better solution. The records are linked together into a list by inserting a pointer field "next" into each record. Each record will then "point" to the record following it in the list. Deleting a record becomes the simple matter of changing a single pointer value as illustrated in figure 2. Given the pointer "current" pointing to the item just before the one to be deleted, the following simple statement will do the trick:

```
current↑.next := current↑.next↑.next
```

Adding a new record is only slightly more complicated.

Let me repeat that these simple examples are not meant to do more than provide a brief glimpse of the marvels of PASCAL's structured types. For full explanations the reader is referred to the texts in the bibliography.

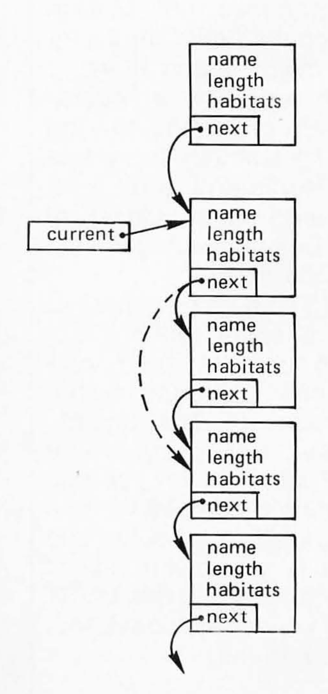


Figure 2: A linked list of records of type "bird" with addition of the pointer field "next." Deleting the third record is a simple matter of changing a pointer field, as shown by the dotted line.

Conclusion

Rapid though it has been, I hope that this survey of PASCAL will have brought out some of the features which make it vastly superior to BASIC. BASIC offers an absolutely minimal set of features and expects you either to devise makeshift solutions or to design a new version of the language when they are inadequate. No wonder there are so many different versions of BASIC. PASCAL offers a somewhat wider selection of features, but avoids the pitfall of trying to include every feature known to humanity. PASCAL is a simple and streamlined language: the *PASCAL Report* defining the language is a mere 32 pages long. Yet PASCAL's designers seem to have chosen just those features which the user needs to expand the language when the need arises, so that it is a genuinely general-purpose language suited to a wide variety of problems. It is this combination of simplicity and power which seems to me to make PASCAL the natural choice for a standard microprocessor language.■

BIBLIOGRAPHY

- *PASCAL: User Manual and Report*, Kathleen Jensen and Niklaus Wirth, Springer Study Edition (2nd edition), Springer-Verlag, 175 Fifth Av, New York 10010.
- *Microprocessor Problem Solving Using PASCAL*, Ken Bowles, Springer-Verlag, New York, 1977.
- *PASCAL News*, Andy Mickel, University Computer Center, 227 Experimental Eng, 208 SE Union St, University of Minnesota, Minneapolis MN 55455.

Languages Forum

A Proposed Pascal Compiler

A Note About the Tiny Pascal Project...

As this special issue went to press, we had just received the first two parts of a 3 part series of articles on a Tiny Pascal compiler by Herbert Yuen and Kin-Man Chung. Watch future BYTEs for this excellent "do it yourself" project for the software experimenter... CH

Herbert Yuen
POB 2591 Station A
Champaign IL 61820

Kin-Man Chung
124 Scottswood Dr
Urbana IL 61801

In the Languages Forum of the April 1978 BYTE, page 150, we read Stephen Smith's report on his *homebrew* compiler project. Actually, he is developing the Pascal subset compiler on a mainframe computer at a university and planning to transfer it to a microcomputer. He said he had a minor problem with code generation (using 6502 machine code). We think his project might progress more smoothly if he uses another approach—that of generating assembly code for a hypothetical stack machine. This is the same method professionals use for implementing portable Pascal compilers on big computers.

Our own homebrew compiler project was developed in house on a microcomputer that uses an 8080 processor and has a North Star disk system. We began in mid December of 1977. Our motivation came from the fact that the North Star disk BASIC, although very good for general programming purposes, was not fast enough for system software development and some graphic games. For instance, our 8080 assembler, written in BASIC, takes 1 to 3 seconds to assemble one single assembly instruction. Assembling a 500 line program takes about one half hour. From various sources of information we know that Pascal is one of the easiest languages to implement. It also has many nice features that are desirable in a high level language.

The Pascal subset is small, otherwise it would be very difficult to develop using a BASIC interpreter. All variables in the subset are 16 bit integers. Arrays are single dimensional. Character strings are declared as arrays and each character takes one array element; although wasting space, this is easy to implement. Procedures and functions may be recursive. Variables and constants, except arrays, can be passed as arguments to procedures and functions. Language statements include declaration, assignment, BEGIN-END, IF-THEN-ELSE, WHILE-DO, REPEAT-UNTIL, FOR-TO/DOWNTO-DO, CASE-OF-ELSE. The subset is big enough to provide useful features. The Pascal compiler can be written in the subset without much difficulty.

The actual coding of the compiler (in BASIC) began in January 1978. The compiler generates P-code for a hypothetical stack machine, the same one described in Wirth's book, *Algorithm + Data Structure = Programs.* (P-code is the intermediate code generated by the Pascal compiler. It is the machine language of a hypothetical Pascal oritented computer. Use of P-code makes the Pascal language portable since only a P-code interpreter needs to be written for a particular processor. This saves the user from writing the entire compiler for each individual machine.) Several instructions and input/output (IO) capabilities have been added. At the same time, an interpreter was also written (in BASIC) to execute and debug the P-code. It helps to verify the correctness of the codes generated by the compiler. In late January, after most parts of the two programs had been debugged, we began to design a run time support package in 8080 assembly language and also a translator that translates P-code to 8080 machine code. With the debug package and

Continued on page 155

Event Queue

August 7-9, Knowing and Understanding Computer Graphics, Toronto CANADA. This 3 day intensive seminar will cover all aspects of computer graphics, demonstrating how effectively and economically they may be incorporated into the business community. Contact Robert Sanzo, Frost & Sullivan Inc, 106 Fulton St, New York NY 10038, (212) 233-1080.

August 7-9, Project Management for Computer Systems, Houston TX. This 3 day seminar will illustrate techniques for planning, implementing, installing and controlling projects. Specific examples and case studies will be discussed. This seminar is intended for computer project managers, data processing managers, VPs of administration, financial managers and others involved in EDP systems development and implementation. Contact the University of Chicago, Center for Continuing Education, 1307 E 60th St, Chicago IL 60637.

August 7-9, Third Jerusalem Conference on Information, Jerusalem ISRAEL. The conference will cover a broad range of topics on computing applications, science and technology. Primary emphasis will be on the role of computers in the transfer of technology between large and small countries. Contact Robert W Rector, executive director, AFIPS, 210 Summit Av, Montvale NJ 07645, (201) 391-9810.

August 7-9, Laser Beam Information Systems, Minneapolis MN. This seminar will cover the growing application of laser technology in image and data manipulation in the form of scanning, transmission, reproduction and control. The principles and practice of laser beam information systems will be covered in preparation for direct application to such fields as facsimile, computer memory and display, target identification, reconnaissance, photocomposition and image manipulation. Contact Philip M Nowlen, program chairman, director, Center for Continuing Education, the University of Chicago, 1307 E 60th St, Chicago IL 60637.

August 8-10, Management Information Systems, Lehigh University, Bethlehem PA. The major objectives of this seminar are to prepare a user manager to better understand and communicate with data processing specialists during the feasibility study, design, conversion, implementation and evaluation of an information system and to equip the manager with sufficient data processing knowledge to identify potential areas for computerization and/or to improve existing systems. Contact Faith Newhall, Administrative Assistant, Industrial Engineering Department, Packard Lab #19, Lehigh University, Bethlehem PA 18015, (215) 691-7000 ext 385.

August 7-11, Coding and Information Theory, University of Toronto CANADA. This course will present the fundamentals of representation, storage and transmission of data. Protection against storage and transmission errors using error detection and error correcting (including Hamming) codes will be developed. Efficiency enhancement through information compressing codes, predictive run encoding and Markov chains (probabilistic finite state machines) will be discussed. Contact Short Course Program Office, 6266 Boelter Hall, UCLA Extension, Los Angeles CA 90024, (213) 825-3344 or 825-1295.

August 15-17, In-house Development of Data Processing Documentation and Procedures Manuals, Lehigh University, Bethlehem PA. This workshop seminar is designed for DP personnel of computer centers who wish to revise or develop a DP documentation and procedures manual. The seminar will allow the individual participants to interact with the lecturers and other participants about their specific problem areas. Some of the program topics will include documentation principles and practice, DP documentation and procedures manual, definition of procedures, the systems development process. Contact Faith Newhall, administrative assistant, Industrial Engineering Department, Packard Lab #19, Lehigh University, Bethlehem PA 18015, (215) 691-7000 ext 385.

August 21-25, Digital Filters, UCLA. This course will provide a practical introduction to the subject of digital filters. Topics will include the frequency approach, Fourier series and integrals, nonrecursive filter design, theory of recursive filter design, discrete Fourier transforms, fast Fourier transform implementation, estimation of power spectra and nonlinear phenomena due to quantizing signals. This course will be of interest to those who use linear combinations of data. The emphasis is on its basic nature and practicability. Contact Nonie Watanabe, Short Courses, 6266 Boelter Hall, UCLA Extension, Los Angeles CA 90024.

August 21-26, Three Short Courses for Engineers, Computer Scientists and Individuals Interested in the Areas of Microcomputers and Digital Electronics, Trenton State College, Trenton NJ. The courses are: assembly language programming and interfacing for the 8080/8085/Z-80 microprocessor, programming in BASIC for the microcomputer owner, and microcomputer digital logic circuits. Each of the courses will cover approxi-

mately the same ground as a normal 1 semester college course and will be presented in the form of an intensive, fully documented lecture coupled with laboratory sessions. Contact The Division of Continuing and Adult Education, Trenton State College, Trenton NJ, 08625, (609) 771-2255.

August 21-September 2, Courses on Microcomputer Interfacing and Analog Signal Conditioning, Virginia Polytechnic Institute and State University. The objective of these programs is to provide an educational experience for scientists, engineers, teachers, managers or technicians in the areas of microcomputer data acquisition, instrumentation, and measurement systems ranging from the analog sensor through the analog data channels to the microcomputer. The courses provide a combined lecture and laboratory experience. Continuing education units are provided for each course. Contact Dr Linda Leffel, Center for Continuing Education, Virginia Polytechnic Institute and State University, Blacksburg VA 24061, (303) 951-5241.

August 24-27, PC '78, Philadelphia Civic Center, Philadelphia PA. The first day of PC '78 (August 24) will be an industry trade show which is open to dealers, the industry and exhibitors' guests. For the remaining three days the full Personal Computing Show and Personal Computing College will be running. Over 80 hours of free seminars are planned. Contact John H Dilks III, Rt 1, POB 242 (Warf Rd), Mays Landing NJ 08330.

August 29-31, Data Processing Operations Management, New York NY. This seminar will offer the senior data processing professional an opportunity to gather the latest management skills. The curriculum is designed toward practical, applied data processing management techniques. Contact Philip M Nowlen, program chairman, director, Center for Continuing Education, University of Chicago, 1307 E 60th St, Chicago IL 60637.

September 6-8, COMPCON Fall '78, Capitol Hilton Hotel, Washington DC. Sponsored by the IEEE Computer Society, this conference will cover computers and communications, interfaces and interactions. Such topics as microprocessors in communications, multiple computer systems, advances in communications technology and many others will be discussed at this conference. Contact Kenneth H Crandall Jr, COMPCON Fall '78, POB 639, Silver Spring MD 20901.

September 11-15, Coding and Information Theory, Georgia Institute of Technology. See August 7-11, University of Toronto, for information.

September 12-14, WESCON/78 Show and Convention, Los Angeles Convention Center and Los Angeles Bonaventure Hotel. Contact Electronic Conventions Inc, 999 N Sepulveda Blvd, El Segundo CA 90245, (213) 772-2965.

September 18-22, Digital Filters, Georgia Institute of Technology. See August 21-25, UCLA, for information.

September 29-October 1, International Microcomputer Exposition, Dallas Convention Center, Dallas TX. This exposition will be directed toward all levels of technology from the professional engineer to the beginning computer hobbyist. In addition to the seminars, a panel of experts will be available to answer questions. Contact Beverly Tanner at (214) 271-9311.

October 4-6, Knowing and Understanding Computer Graphics, San Francisco CA. See August 7-9, Toronto.

October 5-8, Midwest Personal Computing Show, Apparel Center's Expocenter, Chicago IL. More than 200 displays featuring the full spectrum of the latest personal computing developments are expected to be presented by manufacturers and distributors. A comprehensive program of seminars, forums and practical application clinics will parallel the four days of exhibits. Contact Midwest Personal Computing Exposition, ISCM, 222 W Adams St, Chicago IL 60606, (312) 263-4866.

October 16-19, Information Management Exposition and Conference, McCormick Place, Chicago IL. The theme, strategic planning in the information age, was selected to convey the need for a corporate strategy for the gathering, storage, retrieval, dissemination and management's use of information. Computer equipment to be demonstrated will include large, medium and small systems, mini and micro systems, small business systems, terminals, peripherals, data communication systems, data collection and preparation handling systems, magnetic media and accessories. Contact Clapp & Poliak Inc, 245 Park Av, New York NY 10017.

October 24-26, Project Management for Computer Systems, New York NY. See August 7-9, Houston, for information.

October 30-November 1, Second Annual Data Entry Management Association Conference, Sheraton Harbor Island Hotel, San Diego CA. The conference theme is data entry today and tomorrow and will concentrate on the human side of data entry and distributed data entry. There will be a full schedule of seminars, panel discussions, and workshops including hands-on workshops with various vendors' equipment. Contact Marilyn Bodek, DEMA, 16E Weavers Hill, Greenwich CT 06830. ■

Continued from page 10

I see no need for a $20+ kit (eg: Pickles and Trout TVM-04 mentioned in the article) when a $.20 resistor will do. I converted my Philco 9 inch black and white TV set by providing a jack at the base of the first video amplifier transistor and by increasing the value of the emitter resistor of that stage by 100 ohms to prevent overloading from my KIM/TVT-6 interface. The resulting picture is excellent.

Cass R Lewart
12 Georjean Dr
Holmdel NJ 07733

*It is a matter of having a product marketed with coherent step by step instructions for a specific product. While your method no doubt works, you probably have the benefit of personal experience which gives you the confidence to proceed. . .*CH

TRS-80 INTERFACES WANTED

I would like to receive information and schematics on interfacing a Texas Instruments 59 calculator and PC-100A printer to a Radio Shack TRS-80 or information on where I can obtain a board for this use.

I would appreciate hearing from any of your readers who have done this or may know someone who has. Any help will be appreciated.

I am also interested in a S-100 bus adaptor for the TRS-80.

Tom Swalenberg
541 Barnett Rd
Columbus OH 43213

A SOFTWARE EXCHANGE?

At Coloma (MI) High School we have a computer center. In our center we have eight different microcomputer systems plus a 3M Model 5500 test scorer. These systems use four different BASICs as well as a number of different ways of storing programs. The BASICs we use are:

1. PolyMorphic extended version A00
2. IMSAI CPM system BASIC-E version 1.33
3. Altair 8 K BASIC version 4
4. North Star BASIC

The storage systems we use are:

1. Poly 88 Byte Base cassette recording system
2. IMSAI dual floppy disk system with CPM
3. Tarbell cassette recording system
4. North Star minifloppy disk system
5. Standard paper tape

We feel that it is necessary to set up a software library between schools using microcomputer equipment. This would give schools a chance to exchange programs and ideas, and to help other

schools just getting started by sending them already working programs such as games, memory tests, and other material. We can provide programs to any other schools in any of the above formats and would be willing to act as a center to publish all available computer programs for schools, providing other school systems are willing to share in this idea and trade programs. Any interested experimenters who have programs to share with schools are welcome.

Terri Leamer
Coloma Computer Center
Coloma High School
Coloma MI 49038

PET BUG?

Recent PET computers have included a comment sheet stating that Richard Duda's Othello program in October 1977 BYTE, page 60, cannot be run because of excessive subroutine nesting. This is absurd; the program does not contain any subroutine nesting. It does not work as written because PET BASIC does not allow branches out of FOR loops. This probably applies to other versions of Microsoft BASIC as well. On the PET it appears to be acceptable to shorten a FOR by setting the index to its terminal value. At any rate the following fixes work on my PET and should work for other computers (such as Apple II, Radio Shack TRS-80 Level II, etc) which use Microsoft BASIC.

```
01820 J=0
01825 FOR J9=1 TO 8
01830 IF C$(J9)=X$ THEN J=J9: J9=8
01840 NEXT J9
01850 IF J=0 GOTO 1720

02620 F1=0
02630 FOR I1=-1 TO 1
02640 FOR J1=-1 TO 1
02650 IF A(I+I1, J+J1)=T2 THEN F1=1: J1=1: I1=1
02660 NEXT J1
02670 NEXT I1
02680 RETURN
Delete 02690 thru 02720
```

This is an excellent program and the graphics can be easily enhanced to show off the PET's capabilities in this area. A Go board with the pieces at the intersections of the lines is easier to use and more appropriate to the game than a checker type board. It does require an 8 K machine as written, but probably could be compressed.

Some of PET's bugs the new owner should watch out for are that expressions of the form IF NOT X may not evaluate properly, and executing a DIM in the middle of a program may clear all variables to zero. These are real disasters in connection with the PCC version of LIFE, for instance.

Incidently, my PET, delivered April 5, is #13,541. It was delivered less than three weeks after I mailed the order, has worked flawlessly from the start and is a superb machine for the price. Documentation remains less than adequate, so novices beware.

Mike Hughes
POB 393
Rapid City SD 57709

Pascal versus COBOL

Where Pascal Gets Down to Business

Ken Bowles
Institute for Information Systems
University of California San Diego
Mail Code C-021
La Jolla CA 92093

With a few important extensions, Pascal can be an extremely powerful tool for writing interactive business application programs on microcomputers and minicomputers. Pascal provides data structuring facilities generally superior to those of COBOL, and its control constructs allow a systematic and modular approach to program design that reduces development effort and improves reliability compared with BASIC or FORTRAN. The extensions needed make it easy to write interactive programs, use random access (floppy) disk files, handle business arithmetic, and recover from error situations.

A Case Study

In this article we will illustrate the use of Pascal for a program application one might find, with variations, in many small businesses. More general descriptions of the language are contained elsewhere in BYTE and in many published introductory textbooks.

The business we have in mind keeps records of information about transactions with its customers, and also records containing descriptive information about the people with whom it deals. The descriptive records might apply to clients of a law firm, patients of a medical or dental clinic, suppliers of a hardware store with a large and diverse stock, houses currently listed by a real estate firm, users of hardware and software products handled by a computer store, and so on. The transaction records would describe orders for goods to be sold, deliveries, invoices sent, payments, requests for information, promotional literature sent, customer property sent out for repairs, medical tests ordered, etc. Typically each record in the file of descriptive records would correspond to many transaction records. Depending upon circumstances, the transaction records might be stored intermingled with the descriptive records (just as in the shoe boxes that some small businesses now use) or in a separate disk file. They might be stored on the same floppy disk if the files are small, or they might be stored on different disks. In any event, we assume that the number of items in the descriptive file is so large that manual processing of the transactions information represents a significant cost to the business for record keeping. We also assume that the business is small enough that it cannot afford to have its own full time data processing department.

We now consider how Pascal programs written for a small computer might help in the operations of a hypothetical small business, the Zyx Gizmo Store. With many competing manufacturers producing gizmos, it is necessary for Zyx to keep track of many different sizes, shapes, qualities and specialized forms of gizmos. Moreover, the buyer can start with a basic model, later adding modules to obtain a larger and more sophisticated gizmo. Gizmos require periodic maintenance and corrective repairs. Zyx stocks some replacement parts which are installed in customer's gizmos by the Zyx repair department or sold to users who do their own repair work. Some replacement parts are too expensive to stock locally, and Zyx must order them from regional distributors when needed. Gizmos are complicated enough to use that many users require textbooks or short training courses to understand how to use them. Zyx sells the textbooks and runs periodic training seminars for which users pay a small fee. Both the training and repair problems are made complex by the rate at which the technology of manufacturing gizmos is advancing, as new models are introduced by the manufacturers each year. While the similarity of the gizmo to the microcomputer is easily recognized

by many readers, the gizmo model could apply equally well to technology based devices being sold in many fields today.

We can assume that Zyx is large enough to employ several salespeople, repair people, and at least one full time administrative assistant in addition to the owner of the company. In general, when a situation arises requiring communication with a customer, any one of these people may have occasion to refer to the filed records on previous transactions involving that customer. If the customer telephones to request advice about an apparently malfunctioning gizmo, the responding Zyx employee usually needs information about the make, model, size and other details describing the customer's gizmo. If a customer asks Zyx to order an additional module from a national distributor, he or she may call Zyx to inquire about the fate of the order before delivery is actually completed. If a manufacturer of modules for gizmos introduces a new line of devices, Zyx may wish to save on promotion costs by contacting only customers known to be using gizmos compatible with that manufacturer's devices. For these and many other reasons, designated employees of Zyx should have ready access to records on the customer's dealings with the firm. These records make it possible for Zyx to render a personalized service that probably is the main reason why customers come to the Zyx store for their gizmos rather than to a national or regional distribution company.

Of course now that low cost microcomputers have become moderately powerful, it is possible, in principle, for Zyx to maintain its descriptive and transaction records on customers in a floppy disk or small hard disk system. Ideally, the cost of adding a microcomputer to a small business operation is only a fraction of the value received, both in labor costs and in improved customer relations. Moreover, the company could use the microcomputer for maintaining its accounting records, sending bills, keeping track of inventory and so on. We say *ideally* because the effort to write a suite of programs to access and maintain the necessary files can be quite substantial if the programming is done in BASIC or FORTRAN (or assembly language). Using Pascal the effort should be very much less than the equivalent effort using BASIC or FORTRAN.

Since COBOL is becoming available on microcomputers, some comments on COBOL versus Pascal are appropriate. Here the principal issue has more to do with the operating system, within which business programs written in the language will run, than with the language comparison. Given reason-

able operating system support of the language, no one versed in Pascal would consider backing up to COBOL. COBOL's principal attraction in the business computing community has been that it is the most standardized of all the widely used languages. COBOL provides facilities for storing dissimilar types of information mingled together in transaction records intended to be stored in off line media like disks and magnetic tape. Pascal too has very powerful facilities for storing complex data records, and its facilities for building complex programs are far superior to those of COBOL.

Regarding the operating system support, we'll assume in the rest of this article that the user's Pascal program is developed under, and runs within, the UCSD (University of California at San Diego) Pascal Software System (see "UCSD Pascal: A Machine Independent System," May 1978 BYTE, page 46). This system provides what amount to language extensions to Pascal which facilitate the use of Pascal in writing interactive business programs. Some of these extensions will be mentioned at points in the discussion where they are used in our example. The accepted informal standard for the Pascal language, as described by Niklaus Wirth in his revised report on Pascal (*Pascal User Manual and Report,* K Jensen and N Wirth, Springer Verlag, New York/Heidelberg, 1975), lacks definition of several facilities that are really essential if the language is to be convenient for writing business programs. On the other hand, Pascal provides an extremely high level from which these facilities can be added.

Transaction Records

In Pascal, the programmer is required to declare what type of information will be stored under the identifier of each variable. Readers of BYTE should be familiar with the concept of type as it refers to an integer (whole number), real (floating point number), or string (of characters) item stored in the program's memory. Readers may also be familiar with the concept of an array containing a collection of items all of the same type. In effect, an array is a composite type associating one identifier with a collection of many similar data items, ie: all integers or all reals, etc. Pascal allows one to declare one's own composite type containing a collection of items of dissimilar types. Listing 1 gives a concrete example that might apply to the records of the Zyx company.

In Pascal, any type declarations one wishes to make must appear in the main program or in a block (subroutine) before any variable identifiers are declared following the reserved word var. In the example

above, representing part of a block, the variable identifier *inrec* is to be used for temporary working storage of a customer record read in from an external device such as disk. *outrec* is to be used to collect several data items together before writing out to the external device. Both variables are declared to be laid out in memory according to the type declaration for *customer*. In other words, the declaration of *customer* describes the various fields of information that will be found in any record of that type, whether currently stored in main memory or on an external medium.

The first field within a record of type *customer* is a name consisting of up to 30 characters. The name is of type, *string*, which is a UCSD extension of the standard Pascal concept of a packed array of characters. The type *string* is really just a predeclared record type within standard Pascal. In addition to the packed array of characters, the record also contains a single byte field representing the number of characters currently containing useful string information. In UCSD Pascal, a variable of type *string* with no reference to the maximum length (like the *[30]* in the *name* field) will be given a default maximum length of 80 characters. Characters are ASCII and are synonymous with the concept of 8 bit bytes.

The identifier *chargesunpaid* is an extended precision integer represented internally as a 32 bit binary number and limited to storing numbers with up to eight decimal digits of precision. Associated with *chargesunpaid* is a scale factor of two decimal digits, designed to represent dollars and cents. Both the extended precision concept and the decimal scaling factor are UCSD extensions to standard Pascal intended particularly for business use. Where no precision or scaling factor is mentioned in the type portion of an integer declaration (as with the fields *areacode, prefix* and *extension*), the system assumes that the programmer wants the standard integer precision on the machine being used. On most microcomputers this will be 16 bits, equivalent to about 4.5 decimal digits.

telephone is the identifier of a field within the *customer* record layout, where *telephone* is itself a record containing three fields, each of which is an integer. Depending upon the purpose one might have in mind for the data on telephone numbers, it might be better to represent the telephone number field as a string of ten characters. We have used this representation mostly as an illustration of the language facilities.

address is also the identifier of a field which is itself a record containing three fields. Both *telephone* and *address* are said

```
type customer =
    record
        name: string[30];
        chargesunpaid: integer[8:2];
        telephone:
            record
                areacode: integer;
                prefix: integer;
                extension: integer
            end;
        address:
            record
                street: string[40];
                citystate: string [40];
                zip: integer[5]
            end
    end {customer};

var
    x,y: real;
    i: integer;
    inrec, outrec: customer;
```

Listing 1: User declared composite type declaration in Pascal. In Pascal, the programmer is required to declare what type of information will be stored under the identifier of each variable. Examples of standard predeclared types include integer and real. Pascal allows one to declare one's own composite type containing a collection of items of dissimilar types. In this example, the type "customer" has been created, consisting of a record of the variables name, chargesunpaid, telephone and address. String is a predeclared composite type provided by UCSD's Pascal system.

to be "nested" inside the record of type *customer*. Pascal would allow us to nest record type fields within either *telephone* or *address* if we wished to do so, and those record fields could in turn contain other records. In this respect Pascal and COBOL are similar, though the Pascal facilities for record declarations are generally more flexible. As in COBOL, one can declare that a particular transaction record may be used with several distinct field layouts, allowing a file to contain records with several different formats.

In Pascal, one refers to a complete record by its identifier alone. We could transfer the entire content of *inrec* to *outrec* using the statement:

outrec := inrec

No concept similar to COBOL's MOVE CORRESPONDING statement is available to allow the transfer of similarly named fields between records declared to be laid out differently.

If we wish to refer to a single field of a Pascal record, it is necessary to name both the record identifier and the field identifier. Thus we might assign a value to the *name* field of *outrec* as follows:

outrec.name := 'John Q. Public'

In the situation of complex record types with many nested records, one can often simplify the extra writing needed to refer to all the nested record identifiers by using the Pascal *with* statement.

Interactive Input and Output

Input and output (IO) is the area of greatest importance in business applications where the standard Pascal definition lacks a few essential features. Standard Pascal input and output *do provide* an orientation similar to some implementations of COBOL in that a file (an IO device) has an associated buffer variable of the same type as that of the file itself. In the next section we'll consider files associated with record types.

Published discussions of input and output in Standard Pascal are generally limited to handling files of type *char*, meaning that input and output are assumed to consist of a stream of characters. The standard identifier *text* is a convenient way to declare a file identifier as in:

fid: text;

which is equivalent to:

fid: file of char;

The standard Pascal read and write statements provide automatic formatting of external character strings representing integer or floating point numbers into and from their corresponding internal integer and real representations.

While the concept of type *text* is useful when working with magnetic tape devices or with card input and line printer output, it has proven difficult to use with interactive devices. The UCSD Pascal system is extended for this purpose. The principal problem with type *text* for interactive files is the standard Pascal definition of the *read* statement. *read(fid,x)* is equivalent to:

x : fid ↑;
get (fid)

in which the content of the buffer variable is first assigned to the variable *x*, following which a new character is loaded into the file's buffer variable from the external device. This is inconvenient when one would like to place a prompting message on a video display screen, using a simple write statement, following which the program should wait for input demanded by a read statement. The standard mechanism implies that the system looks ahead for a character to be loaded into the buffer variable. This is a great idea for tape files, but not at all convenient for interactive devices. UCSD Pascal

extends this concept by associating type *interactive* with interactive devices. Type *interactive* is the same as type *text* except that the buffer variable is loaded from the external device *before* the value in the buffer variable is moved to the program variable. In more explicit terms:

```
var fid: interactive;
    .
    .
    .
get(fid);
x : = fid ↑
```

where the last two lines represent *read(fid,x)*.

UCSD Pascal extends the idea of types *text* and *interactive* by allowing a string to be handled with minimum fuss. On *read(fid,strg)* (or just *read(strg)*, when referring to the standard system file *input*), one types characters at a video display keyboard with each character appearing immediately on the screen. If a character is mistyped it can be erased from the screen and the input buffer by pressing the backspace key. If one wants to erase the entire input buffer for a clean start (with all typed characters wiped off the screen), one presses the delete or rubout key. The read operation is terminated when return is pressed, whereupon one can determine the number of characters actually input into the variable *strg* by using the built-in *string* function *length(strg)*. On output, the *write* statement determines how many characters to send from a string variable using the length field associated with that variable. For example,

```
write('Hello There');
```

and

```
strg := 'Hello There';
    .
    .
    .
write(strg);
```

would both produce the same 2 word message on the output device. As in Standard Pascal, the width of the field of characters sent from the *write* statement can be controlled as follows:

```
write(strg: width)
```

Disk Input and Output

One of the main reasons for using a disk file is to allow rapid *random* access to any selected record in the file. Access to a floppy disk record takes roughly 0.25 seconds, whereas access to a record on a tape cassette or cartridge can take many seconds or more

than a minute. Interactive business processing usually requires files to be maintained on an external medium like disk or tape because the main memory of a microcomputer or minicomputer is usually not large enough to contain a complete file at one time. Random access is almost mandatory in most cases to avoid long waiting times for the people using the computer.

For example, the Zyx company might have a database of customer records in a file *fcust* declared as follows:

fcust: file of customer;

within the variable declarations of a Pascal program. When a customer arrives to ask for information, a Zyx staff member wants immediate access to the record associated with that customer in the disk file. Standard Pascal provides no way to reach the customer's record without sequentially reading many other records: usually starting at the beginning of the file. UCSD Pascal allows one to position the record number pointer of the file using the built-in seek statement, for example:

seek(fcust, recnumber)

Following execution of this statement, the standard procedure call *get(fcust)* would load the selected record numbered *recnumber* into the buffer variable of the *fcust* file. Contents of the buffer could then be altered directly or moved to other variables in the program. *get* causes the record number pointer associated with the file to be advanced to the next record in sequence. If you want to change the contents of the buffer variable and then return the changed contents to the disk record numbered *recnumber* using *put(fcust)*, you would first have to call *seek* again. The *get* and *put* procedures of Standard Pascal are designed with sequential tape files in mind, and they can also be used for sequential reading of disk files. Use of the *seek* procedure as described allows random access to disk files with minimum alteration of the standard language.

Several aspects of disk file handling are very important for simplifying the task of the business application programmer, though not specified as part of the Pascal language. For example, standard floppy disk media are usually partitioned into sectors of 128 bytes each. In some operating systems, such as the Digital Equipment RT11 operating system, a file is made to appear as partitioned into physical records of 512 bytes called blocks (UCSD Pascal system uses this convention). Typically, the record layout a programmer wants to use (such as *customer* in our ex-

ample) does not result in a neat fit with the sector or block size demanded by the operating system. This means that a logical record associated with a record type declaration in Pascal may occasionally be split between two physical records on the disk. The operating system should allow the Pascal programmer to *get* a record from the disk or *put* a record to the disk without concern for this complication. The system should maintain a directory of disk files so that the programmer need not be concerned with the actual location of a file on the disk, but only with the number of a logical record counting from the beginning of the file.

The programmer of a business applications program package needs to have a simple way to cause a program to call for changes in the library of disk files maintained by the program. For example, an obsolete copy of a master file might be removed from the directory, or its directory name changed. The UCSD Pascal system provides these and other facilities to make disk file handling as painless as possible on a small machine.

Keeping Track of Categories of Data

One of the common problems in business programming is identifying people or things with certain groupings or categories in order to simplify the handling of data on those people or things. For example, the Zyx company might want to characterize some customers as primarily oriented to gizmos made by certain manufacturers, such as the Able, Baker, Charlie, Davis, Edwards, Jones and Smith companies. Within the product lines of these companies, Zyx might also want to have ready access to a record showing which selection of all the possible gizmo modules a customer might have. Thus, when a customer makes an inquiry or a manufacturer brings out a new type of module, Zyx staff members could reduce the effort in knowing how to deal with the customer. For example, a printed promotional brochure might be sent only to the customers associated with an appropriate combination of categories.

In virtually any programming language, this problem can generally be solved by storing descriptive strings as additional fields of the *customer* record. However, the strings can take up far more space than one would like (particularly on a minifloppy disk!), and they are awkward to use when you are simply searching through a file for records corresponding to a particular combination of categories. For example, we might want to search the file to identify all customers who own gizmos made by the Able, Jones and Smith companies who also have a particular type of add-on module. (If you are having trouble relating to gizmos, how about S-100 bus microcomputers with a minimum of 16 K bytes of memory?)

To solve the space problems in storing categories information, a standard technique in traditional programming languages involves deciding on a set of codes to represent the various categories. In our simple example enumerating the gizmo manufacturers, we might store a single letter representing each manufacturer, such as A for Able, B for Baker, and so on. But how do we store the information that a particular customer is associated with two or more of these codes? Without a complex indexing mechanism, a random access disk file virtually requires that all logical records be of the same size. Do we provide an array for storing these codes? How long does the array need to be to account for all possible combinations of codes for our customers? Are we willing to put up with inaccurate data on a few customers in order to save large amounts of file space for the great majority of customers? How do we write a search program to go through the file quickly to find all the customers associated with a specific combination of categories? The reader might well pause at this point to consider how to accomplish these tasks with his or her favorite programming language.

The Pascal facilities for handling sets are designed to make program solutions for problems like these as painless as possible.

```
type
    manuf = (able, baker, charlie, davis, edwards, jones, smith, none);
    customer =
        record
            name: string[30];
            chargesunpaid: integer[8:2];
            equipment: set of manuf;
            telephone:
                record
                    areacode: integer;
                    prefix: integer;
                    extension: integer
                end;
            address:
                record
                    street: string[40];
                    citystate: string[40];
                    zip: integer[5]
                end
        end {customer};

var
    x,y: real;
    i: integer;
    supplier: manuf;
    inrec, outrec: customer;
```

Listing 2: An expansion of the Pascal code in listing 1 illustrating the use of sets. The type manuf has been added, which can be associated with a variable allowed to assume only the values enumerated in the declaration. For example, the new variable supplier, of type manuf, may take on the value of any of the items in the manuf list such as able or davis, but no others outside the type.

For example, we might expand the declarations given earlier as shown in listing 2.

We have added the declaration of a new type *manuf* which can be associated with a variable allowed to assume only the values enumerated in the declaration. For example, the new variable *supplier* is allowed to be assigned the value *able*, or *jones*, from the list of enumerated identifiers.

Also declared as a new field of the *customer* record type is *equipment*, a set of members selected from the type *manuf*. If a customer of Zyx owned gizmos made by Baker, Edwards and Smith companies, the following assignment statement might appear in a simple program:

outrec.equip := [baker, edwards, smith]

where the quantity in brackets on the right side is a set constant stating that items are present from the three manufacturers noted. For an interactive business file maintenance program, the record of a new customer showing no association with a manufacturer would most likely be initialized using an empty set constant:

outrec.equip := [ ]

Then, when the customer acquired his or her first gizmo, we might find a statement such as:

outrec.equip := outrec.equip + [edwards]

which would form the union of the old value of the *equip* set with a new set constant value. In other words, *equip* would now have a notation indicating the presence of *edwards* in addition to what was previously noted in *equip*. We could continue adding notations of other gizmo acquisitions when appropriate. In fact this process is likely to assign a value to a simple variable of the set type associated with *manuf*; then that variable would be used elsewhere in the program to augment the noted membership of *equip*.

Pascal's facilities for handling sets are advantageous in many ways. A set is generally stored in memory as an array of binary bits which are made accessible in a special way. In UCSD Pascal, a set is stored as a string of bytes, each byte containing up to 8 bits to indicate whether a corresponding value is present in the set. Only the number of bytes needed to hold the declared number of set members need be stored. If, as is usual, one needs several dozen members in a set for a business application, the space occupied is very little more than the minimum needed. UCSD Pascal allows a set to have as many as 4080 members.

Once the value of a set field of a record has been assigned, it is readily possible to test whether a customer record is associated with a desired combination of members. For example, to determine whether a customer is noted as owning gizmos made by Baker, Edwards or Jones companies, we could use an *if* statement such as:

```
if (outrec.equip * [baker,edwards,jones]) <> [ ]
    then
        begin . . . end;
```

Here the expression within parentheses (on the left of "<>") isolates the members of *equip* falling in the group Baker, Edwards and Jones. The parenthesized expression is said to be the intersection of the value in the *equip* field in *outrec* and the set constant within square brackets. The comparison indicated by <> then asks whether the result of the intersection operation has left any members by asking whether the result is an empty set. If not, then at least one of the three members must be present, and the compound statement *(begin . . . end)* following *then* is executed.

The alternative to this test for set membership would usually be a complex sequence of IF tests in the traditional languages. The set combining and testing operations can be implemented efficiently by the Pascal system. Thus they allow a

The UCSD Pascal system was described in the May 1978 BYTE, page 46. Interested readers can receive a copy of the software for a $200 subscription fee, which includes the software manual. The manual alone is available for $15 postpaid. Order from: Institute for Information Systems, UCSD, mailcode C-021, La Jolla CA 92093, (714) 452-4256. Checks should be made payable to the Regents of the University of California.

Manufacturers Known to Carry UCSD Pascal at Press Time

Terak Corporation
14425 N Scottsdale Rd
Suite 100
Scottsdale AZ 85260

Northwest Micro Systems
121 E Eleventh Av
Eugene OR 97401

Altos Computer Systems
2378B Walsh Av
Santa Clara CA 95050

Prices on all these systems are in the $5000 to $8000 range for the total package purchased new by individuals at retail. For this one gets a high level language system for personal use which is often far superior to the run-of-the-mill traditional time-shared minicomputer or maxicomputer. Rumor (but no confirmation yet) has it that UCSD Pascal will shortly be available on the Heathkit H-11 and other high end personal computing systems.

program to be written more simply and occupy less space. They also make the operations undertaken by the program more obvious to anyone versed in Pascal, thus making a complex program more easily maintainable and bug free.

There's a Lot More

It is not possible to present a comprehensive view of how one uses a language for complex business programming within a short article. For example, we have not described the use of Pascal *subrange* variables, which allow a programmer to state that a variable is permitted to contain only certain declared values. If an attempt is made to assign to the variable a value outside the declared range, the program either terminates abnormally or (if Pascal is extended in a simple way) the programmer may provide a *recovery block* in which corrective measures may be taken. Data validation is one of the most common problems in business data processing. At UCSD, we feel that the addition of a simple recovery block mechanism is essential to allow reduction in program complexity for handling the many exceptional circumstances that show up in business data, without unnecessary interruption of processing.

A Note on Pascal Extensions

Though Pascal does seem to require a few extensions to make business application programming truly practical, the language provides an extremely powerful base from which to work. One of the strengths of Pascal, according to the intentions of its designer, is that it offers all this power in a remarkably simple and self-consistent form. The necessary extensions can be made in ways that generally retain this consistency so as to be relatively obvious to the programmer. We feel that Pascal is by far the best language available for adaptation to interactive business processing on small machines. We would be happy to send further information about how we use the language for business or real time applications to anyone who writes to us.

The questions of whether standard Pascal should be extended, and how, are currently being debated intensely in the international Pascal Users Group. Each special interest community of Pascal users has its own list of extensions considered essential to make the language a practical tool for developing software products in that community. Even the question of what extensions are essential is being debated, since it is possible to use the facilities of the standard Pascal language to create a library of routines to handle the user's special problems in most cases. In general, an implementor should consider extending the language only in cases where the result will be simpler and more reliable or efficient programs.

This article discusses extensions that the author feels are essential for business applications. Other communities with very strong interests in Pascal work with real time applications, development of system software such as operating systems and compilers, interactive systems such as computer assisted instruction, scientific computations, and so on. Of course these communities do overlap substantially. If the essential extensions needed by all these communities were added to the standard Pascal language, the simplicity and self-consistency that make the language so important would probably be destroyed. Therefore, it is very unlikely that an eventual formal standard for the Pascal language will include any but the most widely needed extensions currently under discussion.

This situation leaves many Pascal advocates very much worried that there will be no effective standards for the extended language features needed by the special interest communities. There has been discussion within the Pascal Users Group about the possibility of encouraging development of common interest supersets of the language for specialized uses. Ideally, language standardization is a process which should proceed slowly giving attention to the ideas of all experts who wish to be heard. In practice, the use of Pascal is growing so fast throughout the computer industry that close coordination of the extensions made by many implementors has become virtually impossible. We at UCSD have set ourselves the limited goal of seeking coordination and cooperation on Pascal extensions for system programming (including those for business and real time applications) among a number of industrial firms that seem most active in use of the language, particularly as regards small computers. For reasons associated with their own proprietary interests, these firms will generally be able to cooperate on only some of the most widely used language extensions within their special interest communities. A Pascal language extensions workshop was held at UCSD in July of this year primarily to help bring about this coordination. We intend to continue working as closely as possible with the international Pascal Users Group, and to take guidance from the PUG leadership on extension issues whenever practical. ■

Continued from page 6

stroking of programs or object code for programs. The traditional manual and job shop methods of production of copies of software for distribution are not appropriate when we think of a mass market of 10,000 to 100,000 copies (or more?) of a program distributed via retailers and mail order houses with a retail price of (for example) $9.95.

The Software Distribution Model

Given an identifiable set of computers with sufficiently similar characteristics, software can be marketed and distributed to multiple users.

The "sufficiently similar" characteristics which make a program marketable to multiple users include the formal representation of the software, and the machine readable medium in which the software is delivered. The machine readable representation of a program product is always accompanied on delivery by extensive printed documentation. At a minimum this documentation describes how to use the product; in the optimal case it includes details of the actual algorithms employed. To summarize, the key points of a delivered product are:

- Formal representation.
- Machine readable medium.
- Documentation.

I'll be making evaluations and comments largely on the subject of formal representation from the point of view of the new mass market for software which is developing in the personal computing field.

Formal Representation

The formal representation of programs to be distributed by a software vendor is one of the key choices which has to be made. At one extreme, the vendor could provide extremely machine dependent and configuration dependent low level code for a particular computer system product. At the other extreme, the vendor of software might provide a largely machine independent formal representation in a high level language shared by a number of computers. At an intermediate point between these extremes, especially in an era of mass production of a small number of processor architectures as microcomputer systems, we find the possibility of delivering configuration independent but machine dependent relocatable representations of low level code for a particular microprocessor instruction set.

For that class of software products supplied by the original manufacturer of a particular computer system, there is no problem providing compatible software at whatever level of representation is chosen. The manufacturer of a system after all controls the detail choices with respect to processor hardware, system configuration and systems software. Since all the details are decided by the particular design, it is even practical to market software in the form of a memory image at the lowest level (possibly in read only memory parts). Since the choice of processor is well defined, the manufacturer can also provide modules of software represented as relocatable machine code, along with a suitable loader program which is part of his systems software. Since the detailed choice of high level language processors is well defined, the manufacturer can also provide applications and systems programs represented in *his* or *her* high level language. The manufacturer of computer systems products at most must deal with a small integer number of processors and high level languages.

We find this model of software delivery by the manufacturer of a system throughout the computer industry to date. Every mainframe and minicomputer comes with low level representations of systems software and (eventually, if not at introduction) with user

Articles Policy

BYTE Publications Inc is continually seeking quality manuscripts written by individuals who are applying personal computer systems, designing such systems, or who have knowledge which will prove useful to our readers. For a more informal description of procedures and requirements, potential authors should send a self-addressed, stamped envelope to BYTE Authors' Guide, 70 Main St, Peterborough NH 03458.

Articles which are accepted are purchased with a rate of $45 per published page, based on technical quality and suitability for the intended readership. As to articles appearing in BYTE magazine, each month, the authors of the two leading articles in the reader poll (BYTE's Ongoing Monitor Box or "BOMB") are presented with bonus checks of $100 and $50. Unsolicited materials should be accompanied by full name and address, as well as return postage.■

Circle 43 on inquiry card.

libraries of high level and low level programs applicable with the particular systems. At the lowest end of the personal computer spectrum of functions we find a similar case: the major programmable calculator manufacturers with their independent incompatible systems provide users with libraries of magnetic cards or read only memories expressed in a form consistent with the particular machines.

But a characteristic of manufacturers of computers is already evident again in the personal computer world, just as it previously existed in the world of minicomputers and larger computers: whatever the resources of the manufacturer, there is no way it can cover all the myriad applications possible for its computer. To draw an analogy from music, we hardly expect a piano or organ company to supply sheet music ("software") with the musical instrument which is suitable for every user's tastes. The music "user" purchases scores according to personal likes. A personal computer provides an analogous opportunity to exercise tastes in software characteristics. Even for the traditional high priced computer, customization through software is for the most part independent of the manufacturer once the basic operating system and software tools have been defined.

In software, the past has seen a large number of custom software vendors grow large in the niches of large scale computing and minicomputer technology. As the number of people using personal computer systems increases due to the low price of these systems, independent software publishing seems to be one of the most promising ways to assure a wealth of options to the user, provided that the difficulties of the N-representation problem can be overcome.

The N-Representation Problem

For the moment, let's ignore all reference to the problem of machine readable data compatibility and simply look at the user's point of view with respect to software. The user has purchased computer X for use in personal or professional contexts. When he or she has made the commitment to the system, our user can in general expect to be able to conveniently load programs created on other X systems from the same manufacturer. But what if he or she wants to load a program created by a neighbor on computer Y from another manufacturer? Or if the user wants to load a program from an independent software vendor? The variety of representations available in the traditional world of computers as well as the personal computer world is large — even within the framework of nominally machine independent high level languages.

Confining ourselves just to machine

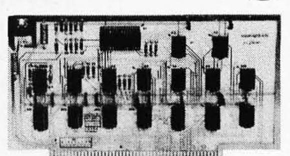
Circle 91 on inquiry card.

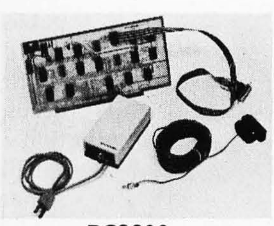
dependent microcomputer assembly languages, there is a wide choice of architectures. At present we find the 8080, Z-80, 6502 and 6800 dominate personal computer architectures. Over the next two to three years we will find added to this list the 9900, 8086, Z-8000 and 6809. If the user of a personal computer sees a neat application system which only comes represented in 8080 code when he has a 6800, that user is effectively unable to run it without a recoding effort. (But even confining ourselves to assembly languages of the same machine design, there is often incompatibility. One vendor of Z-80 software provided an assembler using a hybrid extension of 8080 mnemonics, while others use Zilog Z-80 mnemonics. So the same processor has at least two low level languages available.)

Turning to high level languages, the machine independence of software becomes much greater. But current practices in the personal computing industry are far from machine independent. There is a de facto standard BASIC interpreter in existence, available on most 6502 and 8080 or Z-80 systems. This standard high level language is that defined by the Microsoft company. Extensions and changes of detail accompany each implementation, especially when a given computer has specialized graphics capabilities not available on all the other computers. With the Microsoft design, the major portions of an extended BASIC are identical over a large set of machines.

But Microsoft BASIC is not the only interpreter in existence. A very prominent BASIC in terms of the number of users employing it as represented in the unsolicited articles received at BYTE is the North Star BASIC interpreter. This interpreter is widely used on 8080 and Z-80 systems because of the wide availability of the small floppy disk systems manufactured by that firm: buying a North Star disk peripheral for an S-100 bus system gets the user a limited operating system and the North Star BASIC. The North Star BASIC interpreter and the Microsoft interpreter are inconsistent in a number of fundamental ways in areas of string handling and array dimensions. And these are but the two most prominent interpreters as seen from my point of view as editor of BYTE. I could almost comment that manufacturers take any random formulation of a language vaguely resembling BASIC as originally implemented at Dartmouth, and call it BASIC for marketing reasons. (The temptation to add or delete "features" in a language is of course not confined to BASIC alone.)

From the point of view of a software publisher, the economies of scale obtainable from a mass market will only be obtained if we use a common representation for applica-







tions and systems programs which can be correctly executed by any low level architecture available in the marketplace. With a large number of mutually incompatible software systems, this is not the case. It is my contention that the N-representation problem can be solved *once* by use of appropriate intermediate language representation and efficient interpreters for particular microprocessors. Then the key part of an application or systems program product is the high level language documentation, the equivalent lower level intermediate language object code, and the user documentation: all of course independent of the final machine upon which the software will run. The only machine dependent part which needs to be published is the intermediate language interpreter for a given machine and system configuration. This machine dependent part needs only one definition and one publication version.

Given an interpreter definition, the standard high level language, and the standard intermediate language representation of programs, the user can be assured that once the object code is in place in his machine, the program will run with the same characteristics as described in the documentation for a radically different machine. (Hardware differences due to favorable number representations will make differences in precision and accumulated numeric error effects of course.)

Ruling Out BASIC

To the software publisher, a choice of a high level language and intermediate representation for executable code presents a moderate problem. The widely used BASIC interpreters could be used for a perfectly functional representation for the code of many programs. But such interpreters suffer from many inherent disadvantages:

- Lack of uniform representation.
- Slowness of execution.
- Archaic nature of BASIC.
- Lack of a compact machine independent compiled form.

I've already commented on the lack of uniformity in the various BASIC implementations. The slowness of execution is inherent in this type of interpreter. In extreme cases an active search through memory for a label op code is used to find targets of subroutine calls or unconditional transfers. At best there is a level of semantic interpretation necessary to convert a condensed version of the source code into executed code. Many applications and systems programs cannot tolerate the lack of speed inherent in such interpreters. But BASIC can be *compiled* instead of *interpreted*, so this argument alone is far from sufficient to rule out BASIC.

More important, a language like BASIC as presently implemented reflects an earlier state in the evolution of computer languages, circa the early 1960s, with innumerable ad hoc patches and fixups to add "features." Through the 1960s and early 1970s advances were made in the concept of what a computer language should be in order to be convenient to use and conducive to error free thinking and programming. (For just one contrast, consider this: where the BASIC programmer is required to go almost to the machine language level of assigning numbers to locations in a program, good

contemporary high level languages such as Pascal and its relatives allow the programmer to use meaningful names based on the application being programmed.)

Finally, BASIC as implemented in most cases suffers from the lack of a compact externally available machine independent version of the compiled form of a program. This is an important requirement for the software publisher, since executable code must always be supplied in some machine readable representation, and compactness of representation is important if the inconvenience of relatively slow input techniques is not to discourage the user.

For the reasons just summarized, BASIC is not the ultimate form in which programs are best published. But if BASIC is not the personal computing representation which minimizes the N-representation problem, then what is a better choice?

Enter Pascal

My own personal interest in Pascal came about for reasons which I summarized in the December 1977 BYTE, page 6, in an essay entitled "Is Pascal the Next BASIC?" In this issue several excellent articles including those by Ken Bowles, Chip Weems and Allan Schwartz provide further rationale by way of tutorial argument and example.

This personal viewpoint with respect to Pascal is that of a *user* of a personal computer system who wants to *conveniently* and quickly implement applications and systems software projects ranging from the sublime to the ridiculous. In the sublime category, I include systems software as an art form in itself. I also include writing systems software for my pet projects in musical applications of computers, sophisticated games, and some experiments in the exploration of artificial intelligence concepts. In the ridiculous category, I include such mundane tasks as trivial games, income tax calculations, personal mailing lists of friends and relations, etc. The point about Pascal to be made here is that it is a language well adapted to the utility of computing, whatever your personal definition of utility is. In the range of applications I expect that the Pascal approach to structured, self-documenting, machine independent code will suffice with only an extremely rare necessity to resort to ad hoc kluges in the name of time or memory space efficiency.

From general reading I knew that a Pascal compiler was available and easily transferable to new machines through the use of the technique of "P-code" intermediate language representations. This availability throughout the academic world was one of the reasons for the spread of Pascal, for it is one thing to extemporize about the virtues of a representation and another thing to be able to actually write and examine the properties of code in that representation. Since the original Pascal compilers from Jensen and Wirth et al in Zurich were written in Pascal, producing a P-code intermediate language output file, the task of making the compiler run on a totally new machine architecture was reduced to a relatively simple task of writing an emulator for the hypothetical "P-machine" which executes "P-code" as its machine language.

What I did not know at the time of my earlier comments in these pages is the extent to which that P-code technology had already been applied to small computer systems, in

particular through the work of the people at the University of California at San Diego (UCSD). The UCSD Pascal project has created a nearly machine independent low cost operating system which includes Pascal as the principal high level language, all the usual disk filing system features, support of high resolution bit map graphics including user definable font storage for the character set, an advanced cursor oriented text editor, and interactive compilation and editing features. All the systems software in this package is written in Pascal with the exception of the P-code interpreter and associated detail hooks to the hardware.

The hardware dependent core has already been implemented and is readily available for LSI-11, 8080, Z-80 and 8085 processors. (The cost is only $200 for individual orders, with UCSD quoting a $10 royalty per copy to manufacturers distributing systems in the highest volumes.) At this writing, in the small computer arena, three systems are available which come with UCSD Pascal as a key feature: an LSI-11 system packaged by Terak Corp and heavily used at UCSD, an 8085 processor in a elegant wood finish package with dual floppy drives manufactured by Northwest Microcomputer Systems, and a compact Z-80 system with dual floppy disks manufactured by Altos Computer Systems. Individual users who have 8080 floppy disk systems with the CP/M operating system and enough main memory get a floppy disk to bootstrap UCSD Pascal.

A Serendipitous Result

The nature of the implementation of Pascal compilers, and the UCSD Pascal in particular, leads to an important byproduct: by simply using the UCSD Pascal compiler as the mode of expression of applications programs to be published, it is possible to provide a compact, machine independent representation of programs which greatly simplifies the N-representation problem for the independent software distribution house. The intent of discussing this serendipitous result in print at all is to show the way in which such independent software houses can indeed solve one of the thornier issues and provide their customers with programs which are compiled once yet will run on any one of a number of personal computer systems.

What do we have which already exists in a form which can be readily adapted to a number of small computers? We have the work at UCSD which has produced P-code interpreter based systems for LSI-11 and the family of microprocessors inspired by the 8080 (8080, 8085, Z-80). By the end of the

summer of 1978, indications are that UCSD will also have bootstrapped the Pascal compiler to run on 6502 and 6800 architectures. Taking this P-code interpreter as the input, it is not that difficult to conceive of a self-contained software system which will run in a 16 K byte or larger personal computer system and will contain the necessary interactive user interfaces to load and run a program expressed in the P-code intermediate form as output from the Pascal compiler, but without the necessity of having the full UCSD system available locally to support local compilations.

As a means of demonstrating this concept, a student at UCSD will spend some time this summer creating and characterizing a system based on the UCSD P-code interpreter software for two different machines. This stand alone system will run in the typical current memory sizes of 16 K to 24 K found in personal computers. The goal is to demonstrate a system which can read in a P-code object file (possibly in bar code or audio format), then execute the object file. Issues to be addressed are those of designing the details of the program so that its machine dependent parts can be relocated easily, and so that initial patches for input/output (IO) conventions can be created without excessive mental effort. The machine independent part of this stand alone operating system will be written in Pascal.

In principle, expanding this work to a greater number of processors, it is possible to create a set of Pascal P-code machine emulators which can be published once and only once for each common machine architecture and personal computer manufacturer's configuration, so that this "virtual machine" can be used by a whole family of independent software vendors as a target machine for their wares, rather than requiring each software vendor to solve the N-machine problem separately. By inexpensively publishing the code of the P-machine emulators, we hope to help kindle both an interest in Pascal as a source language and a chain reaction of simplification in the software conventions which must be addressed by independent software vendors. Only time will tell whether or not we accomplish this goal.

A Solution to the N-Machine Problem

Given the existence of such inexpensive standard emulators for the P-machine which executes P-code, a number of beautiful effects become evident for the distribution of application and systems software among a large number of users.

First, since P-code is conducive to use of Pascal as a source language, there will be

a trend toward use of Pascal to express algorithms — a result which is laudable on the abstract and practical grounds of Pascal's beauty as a self-documenting and structured representation of programs. (We already see this trend with respect to BYTE articles presently queued for publication in the near future.)

Second, the N-machine problem of distribution is solved by the device of using the P-machine emulator for each of N-machines as the only machine-specific program, and widely publishing the emulators at as low a cost as possible.

Third, the P-code object code form is a semantically compact representation which in fact minimizes the number of bits necessary to communicate a program to the system which the end user employs. (Yet it maps directly into the source code expressed in Pascal as part of the documentation of the program product in place of flowcharts or other devices.) This consideration is important to the relatively slow IO devices such as FM subcarrier broadcasts of programs, printed bar code copy of programs, audio channel recording of programs, phone network transmission of programs, or silicon real estate of read only memory parts (as inspired by Texas Instruments' SR-59 "Solid State Software" and hinted at by every other semiconductor manufacturer interested in distributing computers at retail).

Why Not Publish Machine Readable Source Code Instead of an Intermediate Language Representation?

The intention of this argument is to provide a way for *compiled* code to be distributed for use with systems which have diverse microprocessor architectures and detail implementations. A key to publishing software inexpensively is the requirement that every detail copy of the software published be identical. Further, a certain definition of the "lowest common denominator" of the set of systems is required.

One way of publishing which is guaranteed to be amenable to a wide variety of detail representations is to publish the machine readable source code of software. But the sheer volume of the code for a well documented source listing argues for a way which is more economical of the user's time and energy. By publishing the machine readable but machine independent intermediate language object code compiled from a printed source listing (also part of a product), the executable representation can be loaded into the machine much more quickly; for program representations in read only memory which are mass produced, an intermediate code representation is also favorable because of compactness relative to source code.

To summarize, the intermediate language approach provides the benefits of *machine independence* coupled with the compactness of representation inherent in the usually *machine dependent* object code for a particular architecture. (The negative side of using a machine independent representation is of course the time overhead of the required low level interpreter. But for a well done intermediate language interpreter, we would expect this penalty could approach a mere 2:1 versus a typical 20:1 or worse penalty for direct interpretation of the source code.)■

Some Notes About Pascal. . .

As this issue was being prepared, a number of interesting bits of information became available:

- *Ken Bowles reports that one associate of the UCSD Pascal project is using the* microcomputer *based Pascal which the project has created in order to write a P-code optimizer in Pascal. The writing of an optimizer program is not in itself particularly noteworthy, but the fact that this optimizer is being written for Pascal compiler output of a Cray-1 computer shows ample evidence of the relative machine independence of Pascal techniques. Here we find the LSI-11 based Terak machines at UCSD (typically a fully loaded LSI-11 with keyboard, bit map graphics, one floppy drive) being used to write, debug and check out programs for one of the world's largest and fastest computers, the Cray-1. (How fast? Fast enough so that light speed propagation limits in the wires become a nontrivial consideration in the physical design of the machine.) Yet the Cray-1 uses a dialect of Pascal for systems programming, and even has a FORTRAN compiler which uses P-code as its intermediate language.*

- *We note that even the US Defense Department likes Pascal as a replacement for such monstrosities as JOVIAL. Two contracts for further language design efforts on the "Steelman" phase of the search for a "DOD-1" language definition have just been announced, with Intermetrics Inc and Honeywell-Bull being finalists in a language design competition based on preliminary proposals. Much of the content of this language definition is expected to be inspired by Pascal, even if it is not a proper superset of the language.*

- *From the industrial side, Texas Instruments Inc has a version of Pascal which is supported for the 990 series of minicomputers, where "supported" means that it is available for use with their disk systems, marketing people are pushing it at seminars for 990 system users, and a comprehensive manual describing the system is available. The 990 series of minicomputers of course includes the microcomputer version of the processor, which is the TMS-9900, and is one of the logical choices for a serious homebrewer or designer of a custom microcomputer system which must use a fair amount of complicated software. The 990 version of Pascal is probably a little too expensive for the individual to purchase, but it represents a very good investment for a commercial user.*

- *Finally, as we went to press with this issue in mid-May, a standards conference, called by Ken Bowles, was scheduled for mid-July at San Diego. Attendance was expected from the worldwide Pascal community, as well as representatives of major industrial concerns, with the intent of defining a set of "standard" extensions to the Pascal language of the Jensen-Wirth report. We expect to have some comments in a future issue about the major points covered in that standards conference. (Of course, the reason for standards must be properly understood: a language standard provides a reference so that any implementer can flag users about how his particular system deviates from the standard. This philosophy is seen throughout computer technology in areas as diverse as character sets for terminals and FORTRAN IV compilers which use the ANSI standard model. A Pascal standards consensus already exists in the Jensen-Wirth report published by Springer-Verlag, and the purpose of the conference is to define an extensions set that covers the superset of the original language necessary to enhance the practicality of the language in real world situations.)*

Pascal is one of the most exciting developments with respect to personal computing we have seen in recent years. The small computer is finally getting to a point where the professionally oriented individual can afford (at the price of a typical new automobile) a computer with some of the most advanced software development characteristics possible in today's computers. Just as a crank starter can get the engine going on an automobile, BASIC and assembly language can indeed be used to program computers. But if one really wants to use an automobile conveniently, an ignition switch and electric starter are now considered essential. The moral of this little simile is that Pascal is the electric starter of the computer world.■

Continued from page 48

interested in becoming members should write to Patrick at POB 7162, Los Angeles CA 90022.

Association of Small Computer Users

An association of users and potential users of small computers has been formed to provide "a new source of unbiased, user oriented information" on the minicomputer and microcomputer market through available publications. The membership fee is $25 per year, which includes a year's subscription to the association's bimonthly newsletter, *Interactive Computing*, a year's subscription to *Minicomputer News*, and two feature reports from Datapro Research Corp, *All About Small Business Computers* and *User Ratings of Minicomputers* and *Small Business Computers.* In addition, members will be eligible for reduced rates on selected publications. Additional information can be obtained by writing to the Association of Small Computer Users, 75 Manhattan Dr, Boulder CO 80303.

G^2C^3 Computer Group

G^2C^3 has been formed in Mobile AL by a group of personal computer users. They meet the first Wednesday of every odd numbered month. Anyone in the Gulf Coast area is invited to join. Present membership ranges from novices to established professionals, ensuring a wide variety of subjects for discussion. Members often bring their computers to these meetings for demonstrations and presentations on their building techniques. For the location of the next meeting and additional information, call (205) 478-1777.

6800 Users Group

A 6800 Users Group has been formed for the Dallas and Forth Worth TX area. It meets on the third Thursday of each month at 7 PM at 1220 Majesty, Dallas TX. All parties interested in attending are cordially invited to do so. Varied topics of interest to users of the 6800 systems along with tutorials in assembly programming are discussed.

Also of interest to users of 6800 systems is the group's "Ask the Chips" feature, where any questions or comments concerning the 6800 are presented and discussed during the meetings. Worldwide users may correspond at the address listed below; the group will make every attempt to respond with solutions.

For further information contact Charles A Matz, 4114 Avondale, Suite 2, Dallas TX 75219, or phone evenings (214) 522-7130.■

Cartoon by Manuel D Juan



Designing Structured Programs

Structured programming is an attempt to modernize software development and to reduce the side effects that divert so much programmer time from actual programming. The use of structured languages like Pascal promotes good programming techniques.

Chip Weems
Dept of Computer Science
Oregon State University
Corvallis OR 97331

In the early days of the computer industry, the most expensive part of owning a computer was the machine itself. Of all the components in such a machine, the memory bank was the most costly because of the number of parts it contained. Early computer memories were thus small: 16 K bytes was considered large and 64 K bytes could only be found in supercomputers. All of this meant that programs had to take advantage of what little space was available.

On the other hand, programs had to be written to run as quickly as possible in order to make the most efficient use of the large computers. Of course these two goals almost always contradicted each other, which led to the concept of the speed versus space tradeoff. Programmers were prized for the ability to write tricky, efficient code which took advantage of special idiosyncrasies in the machine. *Supercoders* were in vogue.

Fortunately, hardware evolved and became less expensive. Large memories and high speed became common features of most systems. Suddenly people discovered that speed and space were no longer important. In fact the roles had reversed and hardware had become the least expensive part of owning a computer.

The costliest part of owning a computer today is programming it. With the advent of less expensive hardware, the emphasis has shifted from speed versus space to a new tradeoff: programmer cost versus machine cost. The new goal is to make the most efficient use of a programmer's time, and program efficiency has become less important — it's easier to add more hardware.

There are some important observations that should be made concerning modern programming. First, the majority of the cost involved with a particular program centers on maintenance and revision rather than initial development. For example, an average program may take three working months to write but can have a lifetime of up to ten years or more, during which dozens of changes may be needed. These can easily add up to several years of labor.

It is also interesting to note that the largest portion of the time spent in revising a program is tied up in analysis of the existing code by the revising programmer. This is the time needed for the programmer to break *into* a piece of code.

Even in the development phase, the largest portion of time is not usually spent on designing or coding, but on debugging. The actual programming takes up very little of a programmer's time in comparison to all of these other program side effects.

Unfortunately, although hardware has evolved rapidly, software techniques have not followed suit to the same degree, since the first high level languages were introduced. Witness that two of the most popular languages in use today, FORTRAN and COBOL, are relics of the late 1950s.

Structured programming is an attempt to modernize software development and to reduce the side effects that divert so much programmer time from actual programming. The main thrust of structured programming is to shift the emphasis of the development phase to careful design in order to reduce debugging time and increase program organization. In addition, special coding techniques make programs easier to revise. The use of structured languages, such as Pascal, makes programs more reliable by permitting





compilers to build more self-checks into the programs they generate.

This sounds like salvation to companies which have to hire expensive programmers and analysts, but what does this extra effort do for the personal computer enthusiast? Consider that the development of the personal computing industry has paralleled that of the entire computer industry, but in miniature. Early personal computers were small and speed versus space was the most important factor. As costs came down, however, larger memories and higher speeds in processors became common. Programming is a spare time operation for many personal computer experimenters; structured programming can help to optimize the programmer's use of this valuable commodity. The difficulty comes in changing old habits and using self-discipline.

There is an anomaly present in the personal computing industry: fourth generation hardware is the rule rather than the exception, but experimenters are still using software, such as BASIC and line oriented editors, which is based on 15 year old designs left over from the second generation of computer hardware.

We have a chance to learn from the mistakes of the mainline computer industry, where people have been digging their way out of the second generation for several years. We have a fresh, new chance to do things correctly.

Structured Techniques

Structured programming is a collection of coding and design techniques that guide programmer efforts into the production of organized, well formed, reliable programs. Unfortunately, this miracle can only come to pass if the techniques are used properly.

Structured techniques are often referred to as being top-down in nature. This classification is really misleading since, in actual practice, both top-down and bottom-up methods are used. Top-down techniques refer to methods which begin with the overall problem to be programmed and, through successive steps, break it down into smaller, more workable subproblems. Bottom-up methods take the opposite approach by starting at the detail level and concatenating small units to form larger units, repeating the process until the solution for the entire problem is formed. This is the building block approach.

Program development is broken into two phases by structured techniques: design and implementation. During the design phase, which is the emphasized portion, the methods used are primarily top-down in

nature. The implementation phase, however, uses mostly the bottom-up approach.

Analysis of any problem is most logically done in a top-down manner. No one would try to solve a word problem in algebra by taking a group of mathematical symbols and trying to fit them together in the hope that an appropriate formula will appear. Such a formula would probably solve only a few special cases or, if it were general, would probably contain several extraneous terms. Even so, this is exactly what happens when people write programs in a bottom-up style, starting out by writing code and not considering the overall design until later.

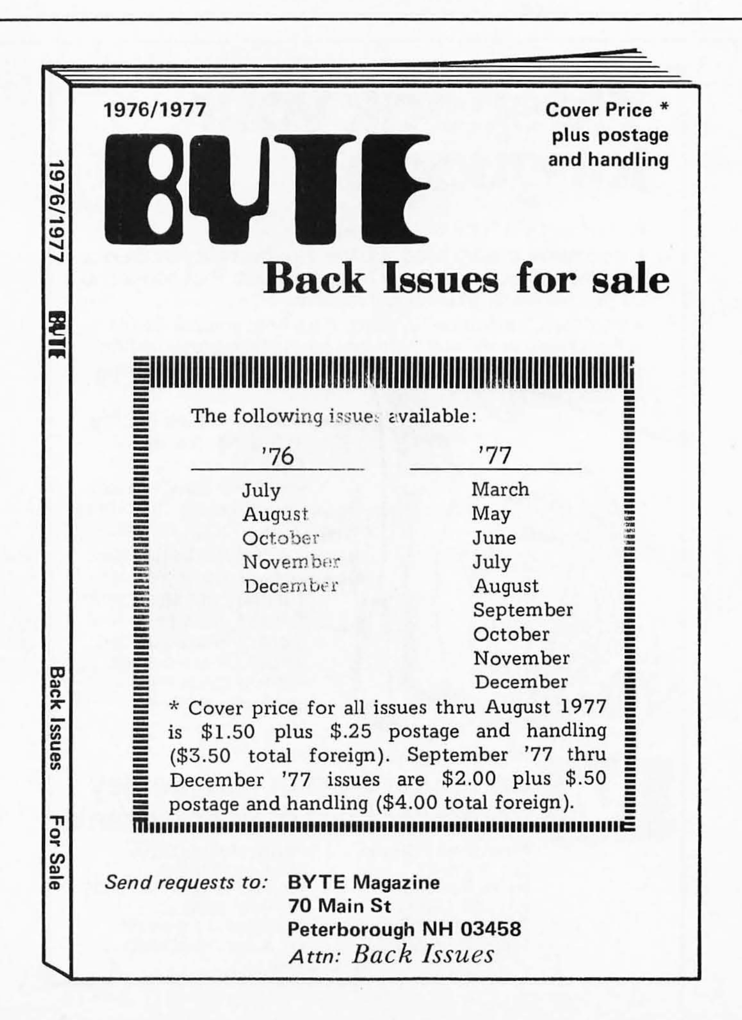
Our first rule in structured programming thus is: sit down at the beginning and analyze the problem in a top-down manner. Break it down into smaller portions so that the overall organization remains clearly visible.

This is the guiding philosophy of the entire design process, the end result of which is a properly coded program. The major techniques used here are stepwise refinement and stepwise decomposition. It should also be noted that this process is not purely top-down in actual use. The experienced programmer knows what is possible and what is not. Such a programmer employs a form of look-ahead along with the top-down techniques in order to avoid impossible designs. Possible coding schemes are constantly being considered while the design is under development. Properly used, this technique can be a valuable evaluation tool and can greatly speed up the design process. The important point is that such a programmer should not get so involved in coding that the top-down approach is completely abandoned.

Implementation is best done through bottom-up techniques. Going back to the algebra problem, we can see that, once we have a well written formula, it would be illogical to try to plug in all of the numbers and do all the computations at once. The best approach is to start with single computations, verifying each one, and build on them until a solution is obtained.

Once the design is completed, the independent, bottom level program modules are implemented and tested first. Higher level routines are built using these subroutines until the program is eventually constructed and the final verification takes place.

Surprisingly, careful design and implementation in this form does not take considerably more time than program development using the older approaches. Some restraint on the part of the programmer is needed, but once the results are seen, it's hard to imagine why anyone would want to continue using the old, unorganized methods.



Circle 95 on inquiry card.

Now that we have a feeling for what structured programming is, let's take a close look at some of the techniques mentioned earlier.

Stepwise refinement is a top-down approach to program syntax. The concept of refinement refers to the language used to describe the solution being programmed. We start with some very general statements about what the program or routine is supposed to do, then work down through several levels of specification using more detailed language at each new level as shown in figure 1.

The end result of refinement is a coded version of the program or module. In order to apply refinement, it is necessary to have previously specified (at least in part) how the program is to be broken down functionally. Another technique, called *stepwise decomposition,* is responsible for determining how the program is split up. It will be examined later in this article. In general, refinement takes place simultaneously with decomposition, but is usually one or two steps behind with regard to level.

Completely different from the end result is the goal of stepwise refinement, which is to reduce the complexity of program development by organizing it into a sequence of manageable steps through the establishment of stable levels of complexity during the design. Once each new plateau of complexity is reached, work can begin on developing the next plateau. This is similar to the way a mountain climber scales a peak: a series of base camps must be established, each one providing a jumping off point for the next day's climb and, in the event of failure, a safe haven to return to for the night. No one would consider climbing Everest in one unbroken effort. There's simply too much mountain there to do it all at once.

Amazingly enough, many programmers still attack mountainous programs with a single effort approach. But there's simply

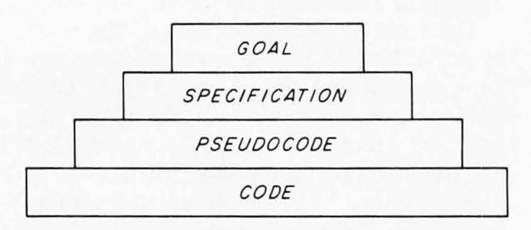


Figure 1: Stepwise refinement of problem. We begin with the high level goal statement, which contains few details. As we step down through each lower level, more details are added until, at the fourth level, we have a coded program or module.

Circle 76 on inquiry card.

too much information in a large program; the human mind can't absorb it all at once. Even though the resulting program may run, its own author probably couldn't explain the whole thing and certainly would not be able to guarantee it to be error free.

How many levels are needed in the development of a program? This depends on the size and complexity of the program, but in general a minimum of four levels is required. These levels are the goal, specification, pseudocode and code levels.

When you decide to write a program, the first thing you should do is write down the goal you have in mind for that program. If you can't write the goal in words, you probably don't have a clear enough picture of what you want to do in order to write a program.

Once you've written the goal, step back and take an objective view. Is it too broad, too grandiose, too narrow, too simple? Writing, "This program will be used to balance my checkbook," is one thing, but, "A program to keep track of cash flow through the entire household," is a completely different matter. The first could be written for a programmable calculator, the second involves establishing a complete data base system.

A great deal of frustration and disappointment could be saved just by writing out the goal and examining it. For a roulette game, a well written goal might read as follows.

"This program will simulate the game of roulette as played by Monte Carlo rules. It will permit up to five players to bet at one time and will use a free form input scheme to simplify the betting process."

In the first sentence, the main goal has been specified and some secondary points have been put down. Note that the goal is not too specific, since the details are supposed to be held off until the specification level. As stated, this program is reasonably difficult, but not impossible.

After a satisfactory goal is set, then comes the specification, which is broken into three parts: input, output, and strategy. Most good programmers have at least once made a statement something like, "If you've got a complete description of what comes into and what goes out of a program, the rest is just adding what runs in between."

Writing a program is, regrettably, not that simple, but it would be much more difficult to write it without knowledge of just how the input data will look or what the output is supposed to be. The specification level provides this and permits us to jot down some rough ideas about how the processing will take place. Table 1 is one possible example for the roulette program.

Obviously, this is not a complete description of the IO, but it is reasonably good. If we were actually writing this program, the specification would be more detailed and several examples would be included. Similar items that can appear in the IO description are card layout forms, disk file formats, record descriptions and so on. If this were a subroutine, our IO specifications would also describe the parameters passed to and from the calling routine.

The strategy is important here, since this is really the second level of the process refinement. Note that there are no system or language dependent statements in the strategy. We should be able to take any task from the specification level of a program description and implement it on any system (with enough resources) without having to change the way we have written it.

Once again, step back and examine what has been done. Is the strategy too detailed, is there enough detail, is it dependent on special system or language features, is it too complex? The last item is important: if the answer is yes, the problem has not been properly decomposed. We must go back to the decomposition and reexamine how we

Input:

Initial number of players: integer between one and five inclusive.
Player names: string truncated at 15 characters.
Yes/no answers: strings truncated to one character.
Bets: string up to 72 characters long of form: integer number bet value *on* bet keywords. All extraneous words and symbols are ignored. Instead of a bet, the special keywords *quit* and *pass* may be input.
Bet keywords: numbers separated by commas.
12H, 12M, 12L (dozens)
12A, 12B, 12C (columns)
low, high (halfs)
odd, even
red, black

Output:

Number of players question.
Players name requests.
Instructions needed question.
Instructions.
Bet requests.
Results and winnings statements.
Bet error messages.
Goodbye message.

Strategy:

After the startup sequence, go into a loop to input the bets. Process each bet by scanning through the string looking for *quit, pass,* or digit. Set the quit or pass flags if either is found and go get the next bet. If a digit, continue picking up digits and converting to decimal until *on* is seen, then scan for keywords. Once the complete bet is processed, check for validity. Either accept the bet or print an error message. Select the random winning number, determine who won or lost and print the results, checking quit and pass flags for special action. Restart input loop if any players remain, else stop.

Table 1: This is the rough sketch of the roulette game which will be used to develop the program. The rough sketch should define what the input and output of the program are going to be and the general workings of the program. It should be very general and give an overall view of the project.

have divided the processing. (Better to backtrack at this stage than after we have gone further and found the problems to be unmanageable.)

In the next phase of refinement we try to get even more specific. On large programs, which may need more than four levels, the extra levels may appear at this point. Any program that requires multiple pseudocode levels has probably been improperly decomposed. A well formed program seldom needs more than the usual four levels of refinement. At this level we may add some system dependent features but we should still try to avoid language dependencies. Our pseudocode description should thus use a wide range of constructs freely, since we can always break these down into whatever constructs our chosen language actually has. Using a wide variety of powerful constructs permits us to think and design more freely, free of the worries of particular language forms.

Many people use a form of ALGOL to write their pseudocode, because the block structure of ALGOL lends itself nicely to structured pseudocoding. At times, this ALGOL type language looks more like a version of Pascal, especially if the programmer is accustomed to Pascal. But the differences are minor because the two languages are similar and a good pseudocoder tries to eliminate language dependencies anyway.

The pseudocode for our roulette game looks something like listing 1. This is still at a fairly high level, but is much closer to being a program than the strategy was. The pseudocode shown uses indentation to indicate blocks rather than **begin** and **end** pairs as in Pascal. This provides a graphical representation of the structure of the program. When indented properly, good pseudocode can be used in place of flowcharts, and is often more easily understood.

A good rule to follow when writing pseudocode is to make it understandable to almost anyone, including nonprogrammers. If you can convince someone that they're going to run a roulette table for people betting via Teletype, they will probably have no difficulty in understanding the roulette program pseudocode. This is an important test. If another person can successfully interpret your pseudocode and, using it, play the part of the computer without running into problems (bugs), then it is well written.

When the pseudocode for a module is completed, it is a relatively simple matter to flesh it out by adding the right words and proper punctuation for whatever language is being used.

Converting our pseudocode to Pascal, for example, would produce something like listing 2.

This is indeed a high level Pascal program. It takes advantage of the ways in which Pascal permits procedures, functions, Booleans and comments to be used, in order to produce actual code that is not much lower than pseudocode. Most of the work done in this program is by procedures and functions defined elsewhere. This is a sign of good decomposition.

```
Begin program.
    Ask how many players.
    For as many players as there are,
        Get each player's name.
    Ask if instructions are needed.
    If yes, output the instructions.
    While there are still any players left,
        For as many players as there are,
            Repeat until a valid bet is obtained:
                Get the player's bet.
                Scan the bet.
                Check bet for validity.
        Determine the winning number.
        For as many players as there are,
            If player quit, process the quit.
            If player passed, process the pass.
            If player bet,
                Determine whether player won or lost.
                Process this accordingly.
End program.
```

Listing 1: Pseudocode for roulette program. This level of design is used to roughly determine what the program should be doing. It should not concern itself with the low level aspects of the program.

```
begin (*program*)
    askhowmany (players);
    for player : = 1 to (*as many*) players (*as there are*) do
        getname ((*of*) player, (*into*) playerlist);
    askif (yes) (*instructions are needed*);
    if yes then printinstructions;
    playersleft : = true;
    while (*there are still any*) playersleft do
        begin (*betting*)
        for player : = 1 to (*as many*) players (*are there are*) do
            repeat
                getbet ((*of*) player, (*in*) playerlist);
                scanbet ((*of*) player, (*in*) playerlist);
                checkbet ((*of*) player, (*in*) playerlist,
                    (*to see if*) valid);
            until valid (*bet is obtained*);
        determine (winningnumber);
        for player : = 1 to (*as many*) players (*as there are*) do
            begin (*processing results*)
            if quit ((*by*) player, (*of*) playerlist)
                then processquit ((*by*) player, (*of*) playerlist,
                    (*updating*) players, (*and*) playersleft);
            if pass ((*by*) player, (*of*) playerlist)
                then processpass ((*by*) player, (*of*) playerlist);
            if bet ((*by*) player, (*of*) playerlist)
                then processbet ((*by*) player, (*of*) playerlist,
                    (*using*) winningnumber)
            end (*processing results*)
        end (*betting*)
end. (*program*)
```

Note: The delimiters (*and*) are used in this listing to indicate comments. Although the Pascal defined symbols are { and }, most output devices do not have these symbols available. Therefore the parentheses and asterisks are substituted.

Listing 2: Pascal program for the roulette game. This is the main program which calls many other subprograms to perform the low level logic. Notice the similarity between this program and the pseudocode of listing 1.

A main program should very seldom have to exceed more than one or two pages in length. If it does, it is probably too complex, and more of its operations should be consolidated into subroutines. Breaking programs up into manageable pieces is the idea behind stepwise decomposition. As I mentioned earlier, decomposition is the other half of the design phase. Now that we've concluded our development of stepwise refinement, let's take a closer look at decomposition.

Decomposition

Stepwise decomposition is a top-down approach to program semantics. Using decomposition, we examine the meaning of a program segment to determine its function with regard to the whole program. Our purpose in doing this is to determine the complexity of the function in order to see if it should be further decomposed. Decomposition refers to the breaking up of program functions to reduce their complexity and increase manageability.

The end result of stepwise decomposition is that all program functions will be reduced to a set of simple, isolated subroutines as shown in figure 2. Like refinement, however, the goal of decomposition is to provide a sequence of levels, like mountaineering base camps, which reduce the design process from a single, giant leap to a series of short, easy steps. Unlike refinement, there is no usual number of levels required to fully decompose a program. The size and complexity alone determine how many levels of breakdown will be needed for a program, be it two or 20. The reason for this is apparent if we note that refinement follows a few steps behind decomposition. With refinement we have the advantage that we will

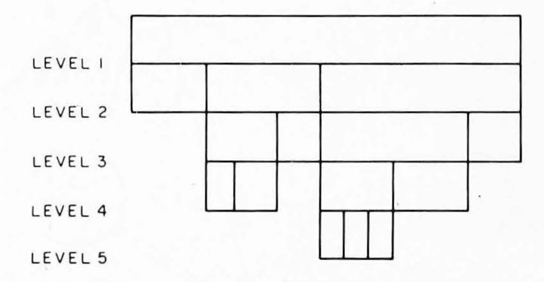


Figure 2: Stepwise decomposition of problem. We begin with the entire problem, breaking it into smaller subproblems. This process is repeated through as many levels as needed, until we are satisfied that all of the resulting modules are as independent and primitive as necessary. Note that not every module has to be decomposed to the same depth.

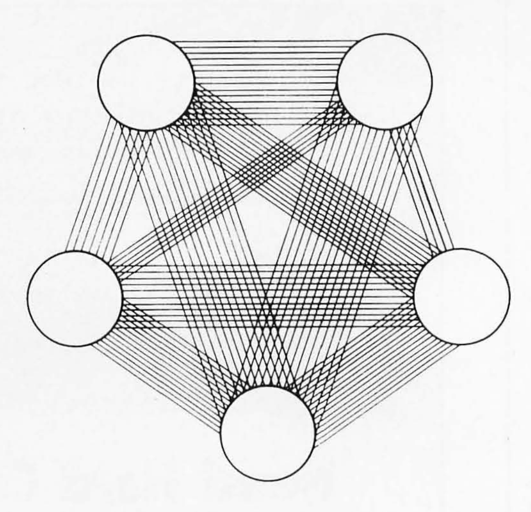


Figure 3: Strongly connected modules. The broad communication paths defeat the purpose of separating functions into modules. In addition, this permits errors caused by program changes to propagate throughout the entire program.

always be working on simple, decomposed program modules which should never exceed a certain level of complexity. Decomposition, on the other hand, is used for breaking new trails and is applied to problems with an arbitrary range of complexity.

Since there are no fixed decomposition levels, we must have some rules that specify under what conditions a new level should be created. Because programming is not an exact science, our rules are not like laws that must be obeyed, but are rather heuristics or rules of thumb which guide us as we work through a design.

Before we can set up these rules of thumb, we must consider two important characteristics of program modules on which they will be based. The first is the connectivity of a module and the second is its functional relationship.

Connectivity refers to the number of channels of communication between a module and the rest of the program. A module which has many links to other parts of the program is said to have high connectivity, while one with few links has a low degree of connectivity.

The level of internal functional relationship is a description of how the processes in a module relate to each other. These processes are said to be highly related if they are all involved in providing the overall function of the module. A group of processes which do not contribute to the module's function is said to have a low functional relationship with regard to the module. The overall level of relationship throughout the module is called its cohesiveness.

Now that the terms and definition have been set forth, let's take a closer look at these two characteristics and see how they relate to structured programming.

Programs whose modules are connected by large numbers of communications paths are obviously more complex than those with only a small number of intermodule links. Modules which depend heavily on other modules, as in figure 3, are difficult to work with because understanding them also requires a complete understanding of all the modules upon which they depend or which depend upon them. This can develop into a chain reaction, requiring the programmer to design the entire program at the detail level—often an impossible task.

Another negative result of high connectivity is that a change in any module can cause effects which ripple throughout the entire program. The numerous, broad communication channels do nothing to restrict the propagation of errors from one module to another. Since the goal of structured programming is to reduce complexity and errors, our design efforts should strive to reduce the number of intermodule connections, as in figure 4.

Modules are usually of two sorts: subroutines (procedures and functions), or blocks of code within a program. They can communicate through shared variables, common data areas, formal parameters, and even by flow of control. (Shared variables and common data areas produce strongly connected programs as in figure 5.) The information communicated via these paths can be divided into two types: that which affects data, and that which affects flow of

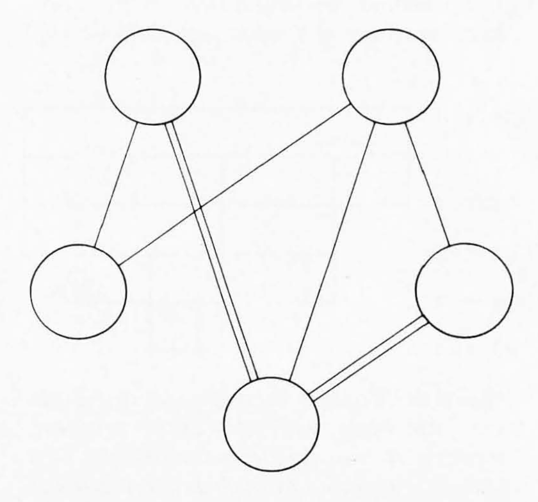


Figure 4: Weakly connected modules. Narrow lines of communication enhance the black box qualities of modules. They also serve to isolate the effects of errors to small sections of the program.

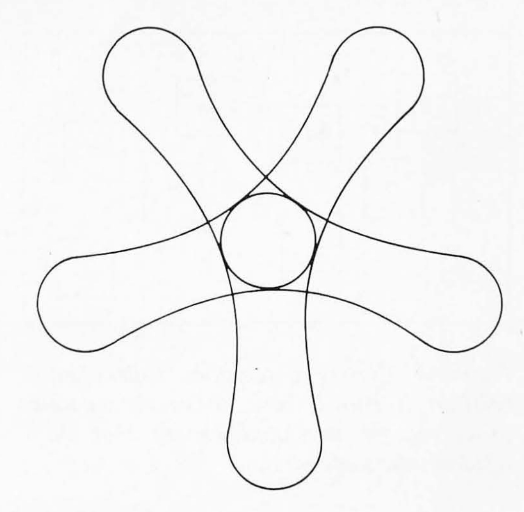


Figure 5: The effect of common or shared data. Each module is strongly connected to every other module which has access to the data, to the point that there is very little isolation. Any action by any module can thus have serious effects on the rest of the modules. Note the similarity to strongly connected modules.

control. The paths themselves can be of two forms: those which connect to a module, and those which connect to something inside a module. The various means of communication are illustrated in figure 6.

The level of connectivity for a path is determined by the complexity of the information communicated, the type of information, and the type of path. Connectivity would be almost impossible to evaluate, quantitatively, but, qualitatively, we can see that some types of connection are worse or better than others.

To be specific, simple connections are better than complex ones, paths which talk to modules are better than those which talk directly to internal processes, and communications that affect data are preferable to those which affect flow of control. By far the worst form of connection is that in which one module modifies the internal code of another, since this requires that the modifying module "know" how the subject module works at the machine level. Thus, any programming changes done to the subject would also require changes to the modifier.

Externally connected paths are rated higher than internally connected ones, since at all times the entire module should be able to supervise any communications. Internal connection bypasses this and makes it impossible for the module to be sure of its internal status at any time. Communications that affect the flow of control require that the sending module have some knowledge of how the receiving module works. Thus, pure data communications are preferable.

In addition, data can be further divided into two types: broad scope and narrow scope data. Broad scope includes such things as global variables, common data areas, and shared data, as represented by figure 5. Narrow scope data is characterized by formal parameters passing between subroutines, which produce less complex connections since these are very specific. In contrast, a change made to one of the broad scope forms can affect all modules connected to it, possibly without their knowing that this has occurred. Broad scope data also adds the problem of determining which module is permitted to modify the common data. Is unanimous consent by all connected modules required, or do we run our program under a democracy? For these and other reasons, broad scope data can be problematical.

Let's summarize what our most desirable configuration would be. All modules would communicate by passing minimal amounts of simple data through well defined formal

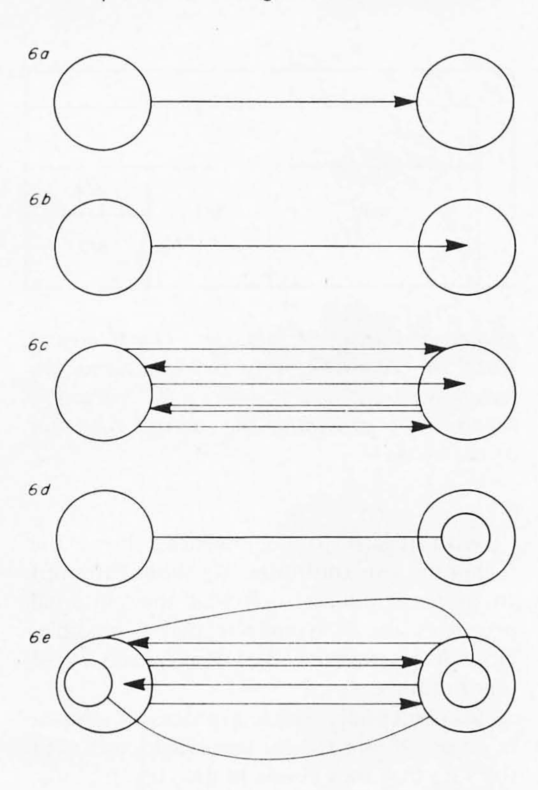


Figure 6: Types of communication. Figure (a) shows simple external connection, which is the most desirable. In (b) we have simple internal connection. Figure (c) is complex communication. With (d) we see communication in which one module modifies the internal workings of another. Lastly, figure (e) shows the worst type of communication: a complex combination of all the other forms.

parameters; in other words, minimal connectivity. Obviously this requires that the modules be broken down so that each provides one simple, dedicated function. Now we're ready for a further look at cohesiveness.

The relationships, which processes internal to a module can have with respect to the module and with each other, have been broken down into six classes. These are listed in order of weakest relationship to strongest.

- Coincidental.
- Logical.
- Temporal (or time-wise).
- Communicational.
- Sequential.
- Functional.

We can see from our discussion of connectivity that it is highly desirable to compose modules such that all of the internal processes are directly related to the specific function of the module. Such processes are functionally related, as illustrated in figure 7.

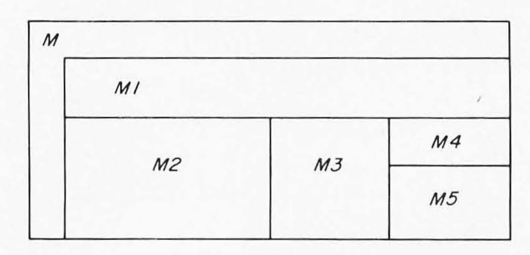


Figure 7: Classes of relations. The strongest form of cohesiveness is the functional relation wherein every process is integrally involved in providing the overall function of the module.

With regard to cohesiveness, then, the higher the level the better. We should attempt to design modules such that their internal processes are as strongly related as possible. Now let's examine the weaker classes of relationships.

Coincidentally related processes, shown in figure 8, are totally unrelated except for the fact that they reside in the same module. This is the weakest form of relation, and any module composed in this way should be broken apart into separate processes.

Logically related processes, shown in figure 9, have no real relation to each other except that they perform similar functions and thus get grouped together to form a module. Once again, this should be broken down into smaller modules.

Processes which are temporally related, as in figure 10, are almost identical to those which are logically related except that, in addition, they can all be executed at one time. A common example is the *initialization* routine found in many programs. Much of what is considered to be initialization by many people could actually be done much later in the program and could be distributed among the modules.

Processes related by communication are those which have been grouped into a single module because they share data in some way, as seen in figure 11. This is a stronger

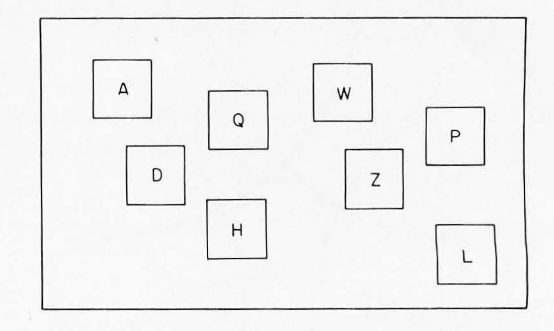


Figure 8: Classes of relations. Coincidental relation is shown here, in which separate processes are unrelated except that they reside in the same module.

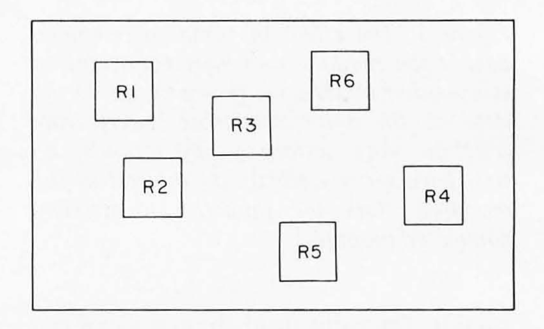


Figure 9: Classes of relations. Shown here is logical relation, in which processes are grouped together because they have similar functions.

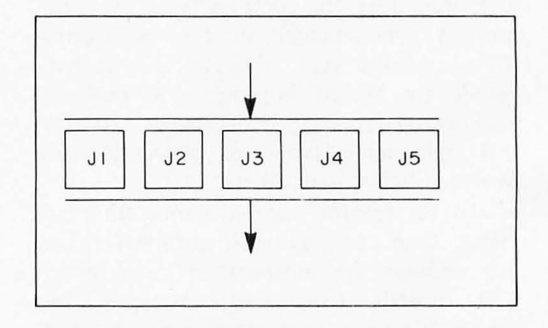


Figure 10: Classes of relations. In a temporal (time) relation the similar processes can execute in any order, effectively in parallel.

relationship, but consideration should still be given to breaking up such a module. A common solution to this problem is a master module which holds the data and formally passes it to each of the submodules as needed.

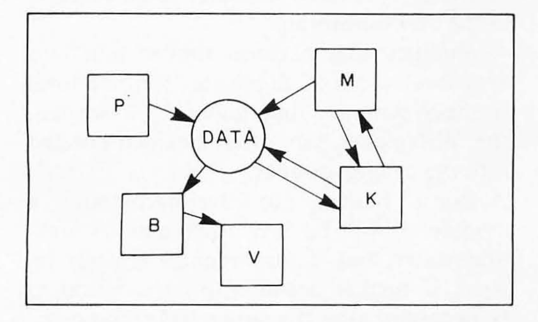


Figure 11: Classes of relations. Processes can communicate with each other directly or through common data in a communication relation.

Sequentially related processes, figure 12, are those which execute in sequence, each one creating data to be used by the next. Obviously such a module can be split apart and the resulting submodules called in sequence with the data passed as parameters. This has the added advantage of making these elementary processes directly accessible to the rest of the program.

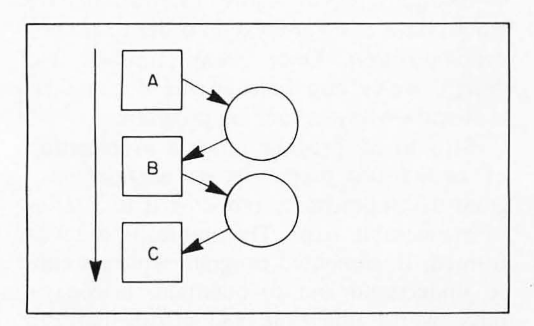


Figure 12: Classes of relations. A sequential relation wherein the first process generates data required for input by the second, and so on.

We can again summarize what our ideal module will look like. It will be minimally connected to the rest of the program and will have a high degree of internal cohesiveness. This means that each of our modules should perform one specific function and should not contain any extraneous processes. In addition, they should communicate solely via simple, well defined, narrow interfaces — preferably by passing formal

parameters. The result is a program composed of plug-in modules that act like black boxes. Such a program is relatively easy to change, because each function has been carefully isolated. Modification then becomes primarily a matter of module plugging, and the effects of any new bugs will be confined to a relatively small portion of the program. Recall that easy program modification and the reduction of side effects were two of the major tasks we were trying to accomplish through structured programming.

Finally, we can list those decomposition heuristics which were mentioned earlier and many of which should be apparent by now.

One good technique for determining whether a module is sufficiently cohesive is to write a sentence describing its purpose. If the sentence *has* to be a compound sentence, the module is probably doing more than one thing. Usually this will be obvious just by looking at the sentence. Incidentally, this can become the goal statement for use in the refinement process.

Carefully examine modules which result in fewer than five or more than 100 source statements of code. This often indicates improper decomposition.

Avoid initialization modules wherever possible. These reduce the black box qualities of many modules.

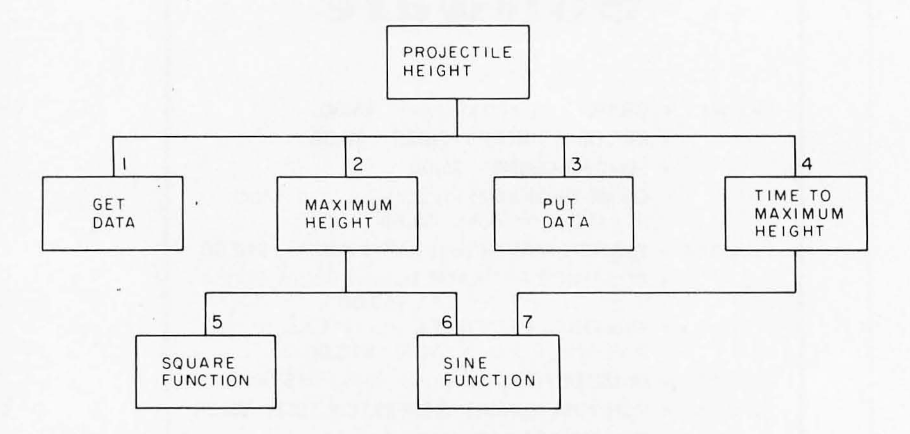


Figure 13: A module structure chart for a simple program which calculates the greatest height reached by a projectile and the time it takes to get there.

	Input	Output
1.	----------	Initial velocity, angle
2.	Initial velocity, angle	Maximum height
3.	Maximum height, time to maximum height	----------
4.	Initial velocity, angle	Time to maximum height
5.	Initial velocity or sine of angle	Square of input
6.	Angle	Sine of input
7.	Angle	Sine of input

Examine your design for duplicate functions. After eliminating these, if any duplicate code remains, it is probably needed.

Watch out for modules which are called by or which call a large number of other modules. This often indicates some problem in the decomposition.

Modules that perform similar functions probably contain duplicate subfunctions. If these common functions can be isolated, the differences can often be incorporated into the calling routines.

Don't hesitate to over-decompose a module. It will be easy to recombine functions later, but it may require a major rewrite if further decomposition is found to be necessary after the design has proceeded.

There is one area in which structured programming truly shines: documentation. If we've done our design properly, there isn't any need for additional documentation.

This may sound a bit outlandish, but consider what our refinements have really produced. The goal statement makes a very fine program abstract; the specification level provides us with a complete technical description, and properly indented pseudocode can take the place of the program flowchart. By appropriately including parts of the pseudocode in the code, as in the roulette example, we end up with a well-commented program. Little else is needed except a chart of the modules and their connections, as in figure 13, but this we would have to do anyway in order to aid our decomposition. Once we've finished the design, we've also finished the documentation and we have a running program.

Structured programming is a collection of techniques that help us organize program development by reducing it to a series of manageable steps. The end result is a well formed, documented program which is easy to understand and to maintain. In today's busy world, the time and effort that can be saved by practicing structured techniques is of immeasurable value.■

REFERENCES

1. Stevens, W P; Myers, G J; Constantine, L L; "Structured Design," *IBM Systems Journal,* 1974, number 2, pages 115 thru 139.
2. McClure, C L; "Top-Down, Bottom-Up, and Structured Programming," *IEEE Transactions on Software Engineering,* volume SE-1, number 4, December 1975, pages 397 thru 403.
3. Wirth, Niklaus; "Program Development by Stepwise Refinement," *Communications of the ACM,* volume 14, number 4, April 1971, pages 221 thru 227.

Continued from page 117

simulator in the system (see Kin-Man's article "An 8080 Simulator" in the October 1977 BYTE, page 70), we did not have much trouble debugging the run time routines. During March most of our time was spent in refining all the routines: revising some features and extensions in the compiler, adding local optimization capabilities in the translator and improving the efficiency of the run time routines. The run time routines, which perform all 16 bit integer arithmetic and logical operations and IO conversions, take only 1 K bytes of memory.

The first step in the bootstrapping process was to write the interpreter in Pascal since it is the slowest but shortest program. It was coded by straightforward translation from the BASIC version. Debugging was smooth and the entire program was up and running within a week. Compared to the BASIC version, the Pascal version runs about 15 times faster; slightly better than we expected. Our next step will be writing the translator and compiler in the Pascal subset. After that, further development can be done in Pascal without the BASIC interpreter.

For three months, each of us have been spending about 10 to 15 hours a week on this project. The first version (in BASIC) of the compiler and supporting software were completed with an estimated effort of two working months. Considering such a short time period and a functioning compiler, we believe we are approaching the task from the right direction. We hope that our project will attract the attention of many readers so that we can share our interest and experience in Pascal with them.■

Languages Forum is a feature which is intended as an interactive dialog about the design and implementation of languages for personal computing. Statements and opinions submitted to this forum can be on any subject relevant to its purpose of fostering discussion and communication among BYTE readers on the subject of languages. We ask that all correspondents supply their full names and addresses to be printed with their commentaries.

Let Your Fingers Do the Talking

Add a Noncontact Touch Scanner to Your Video Display



Steve Ciarcia
Box 582
Glastonbury CT 06033

"Thanks for coming, Steve. I'm glad we were finally able to schedule this meeting. This problem we have is driving us crazy." Fred scurried over to me in the waiting room and shook my hand.

"Let's get you signed in at guard headquarters and then I'll introduce you to Ted."

This was the first time Fred and I had ever met. But his look of relief told me he thought I was some kind of engineering whiz kid. I picked up my briefcase and we walked to the guard's desk. The place was the prototype for a blue chip company waiting room. Decked out with numerous perfectly blended chairs and sofas, it gave the impression of slick tastefulness, and above all *money*. Current issues of various news and business magazines were arranged neatly on the highly polished end tables. I imagined somewhere within the inner depths of the company walls a heavy walnut grained office door with a brushed brass plate reading "Customer Coordinator for Waiting Room Impressions."

My "life in a big company" fantasies were interrupted as I signed my name to the guest card. Signing my name and title was the least significant thing they had me do. There were questions of citizenship, social security number and sex, statements that I represented an equal opportunity employer, and a list of subversive organizations to which I might belong. The urge to check them all and watch the bells and whistles go off was curtailed by my basic marketing instincts.

I passed the card, which ultimately revealed more information than even my wife knew about me, to the guard. He frowned and scrutinized the card carefully. The delay was agonizing as he examined every detailed answer.

"I'll have to inspect that briefcase, buddy," he said.

Surely I'm no buddy of yours, sir, I thought to myself. I fully expected the frown I usually receive when my briefcase is inspected at airports. The inspectors thumb through the piles of paperwork and, upon discovery of an issue of BYTE, quickly cover up this unusually titled magazine and gulp an embarrased "Next!" Much to my surprise the guard seemed uninterested.

"OK, here's your visitor's badge. Remember, you have to be escorted at all times," he said, and whisked me away with a sweep of his hand.

Fred appeared relieved. I was now on the inside and hoped I could help alleviate his urgent problem.

"OK, Fred, what's your problem and how can my company help you?" This was a basic marketing question for our type of business which specializes in technical solutions through custom electronics — which really means providing engineering consulting to companies who have become embroiled in political debate over the latest in-house technical fiasco.

"We'll get to Ted's office in a few seconds and I'll let him explain. Basically we need a black box."

Before I could get the functional requirements from Fred, we arrived at Ted's office. Being introduced to Ted as "director of marketing" elicited a certain degree of respect, because in his company this was a vice-presidential position. Ted motioned me to a seat at his mahogany conference table

near the window overlooking the company golf course. After asking how we wanted our coffee, he stated in a very businesslike manner, "I presume Fred has filled you in on the problem?"

Fred jumped in before I could answer, "I'm sorry, Ted, I haven't had a chance to."

Ted stood up, rotated his body 90 degrees and pointed to the video display terminal in the corner. "That's my problem! Or rather the computer types downstairs who program it!"

I looked at the display. It was a standard graphics terminal similar to those available from several manufacturers.

Ted continued, "Programmers program computers for other programmers! They never think of the user. I drag that terminal to board meetings so we can review marketing figures, and I spend half my time entering 8 digit passwords, hitting escape and control keys to select options, and answering endless quantities of mindless interrogation." Ted was getting a little hot under the collar. "Time is money in those meetings and here in my office. I don't want to spend all day playing true confessor with a computer! Its function is to display information and that's all the interaction I want."

Ted's problem was not unusual. Where a program requires that the next entry be a control R, one had better type a control R. In higher level systems operators need all kinds of cross reference manuals to communicate in the different languages.

"Look," Ted turned on the display and typed the log-on password and terminal identification. Various options were displayed. "This is what I mean. If I want one of these options, I have to type a 5 digit code, wait to give a particular file number and then some other code."

As displays flashed on the screen I couldn't help but offer the obvious question, "Ted, why can't your programmers just change the software to allow single or 2 digit entry?"

"That would be fine if the software weren't already written. We're talking about millions of dollars worth of software and I'm using only a small portion within a large operating system. I want to be able to choose what I want simply."

Ted needed a "black box" and he knew exactly what he wanted.

"I want something to replace this keyboard for the limited specific application of menu selection and display. Put a log-on button on it. When I press log-on it will send whatever information is necessary. The user should know only that he or she has to log into the system—that's all. Next, give me a key that will send the necessary message to

get into the menu programs I use and then I can select the options by number. You send any other messages that are necessary."

Ted was not discussing the usual black box. He was promoting the idea of intelligent rather than dogmatic communication with the computer. A person at this level in the corporate structure could not be expected to maintain the code word and syntax library of the average programmer downstairs. What he wanted was only logical. I left the meeting with the feeling that here was a man who also realized it was time to fight rather than conform.

Perhaps if computers were programmed less for interaction with computer peripherals and more with the human operator in mind, people would be less afraid of them. Ted's application was specific and repetitive but he was still burdened with the general system protocol. In a company that probably had a thousand programmers generating software, his cry to change everything to allow simple input and output (IO) for his application would be fighting an uphill battle. He knew this and also realized that it was easier to change it at his end.

We would make Ted's black box for him and it would solve his immediate problem, but what of the future?

Do your computer input devices limit you? Many personal computer systems have this problem.

Consider a simple program to teach your child mathematics. Such a program in its least complicated form might involve a multiple choice and printout something like this:

```
4 x 8 = 28, 30, 31, 32, or 35
The right answer is?
```

Most BASICs would require typing 32 and a carriage return. Don't forget the carriage return! Remember, you have to conform to the input protocol of the BASIC.

Now, before I explain what I'm driving at, let me give another example. Say you want to use your system for a home management application, such as putting together a shopping list. You could list out the following on the screen:

```
1. Milk          6. Peanut butter
2. Butter        7. Dog biscuits
3. Margarine     8. Cheese
4. Eggs          9. Coffee
5. Rice         10. Tomatoes
   control P for next page
```

Obviously, the number and a carriage return could be entered to choose the items that would be ultimately listed out as a shopping list. A few pages along in the listings, though, the entry data will get more complex strictly from the sheer volume of possible choices. Most homemakers would tire

of the complexity of such a system even though the concept of just choosing items from a list sounds simple.

The solution is to watch the way our young mathematics student might react when we display the expression 4 X 8 on the screen. The natural response is to *point* to the answer!

The homemaker would appreciate using a system that communicates in straightforward terms. Display a list of groceries and let the user point to the desired items.

A New Data Input Device

How do you point to a particular selection on a video display generated menu? The computer needs to know how to interpret your response regardless of the input device. The ASCII keyboard is strictly an input code to the computer. There are unique codes for each switch on the keyboard. The computer doesn't know the location of the particular key that prints an R or a Q. It recognizes only a 7 bit code for these letters. If you don't have a keyboard on your computer, but want to check out some software that needs very little typed entry, you could use seven toggle switches. It would be very slow, but the computer wouldn't care. All it's concerned about is that you present the code it wants.

The same goes for any device attached to a computer. The most obvious way to point to a video display screen and have the computer understand it is to use a light pen. Such units have been described before in BYTE so I won't go into too much detail here (see the references at the end of the article). All a light pen interface does is present to the computer, usually in the same manner as a keyboard input, a code representing a position on the video display screen. This code has to be translated by the program from a position into an action. More on this later.

But, why use a light pen? This again makes the operator conform more than necessary.

Fingers Came Before Light Pens!

Though not capable of the same positional resolution as the light pen, it is possible to design an interface that allows a noncontact data input. Photo 1 is a picture of the prototype designed to illustrate such a technique. It is an infrared scanning system that serves as a low resolution noncontact digitizer. In this particular case it is mounted on the front of a video display to approximate the function of a light pen, but it could just as easily be laid over a typed sheet of

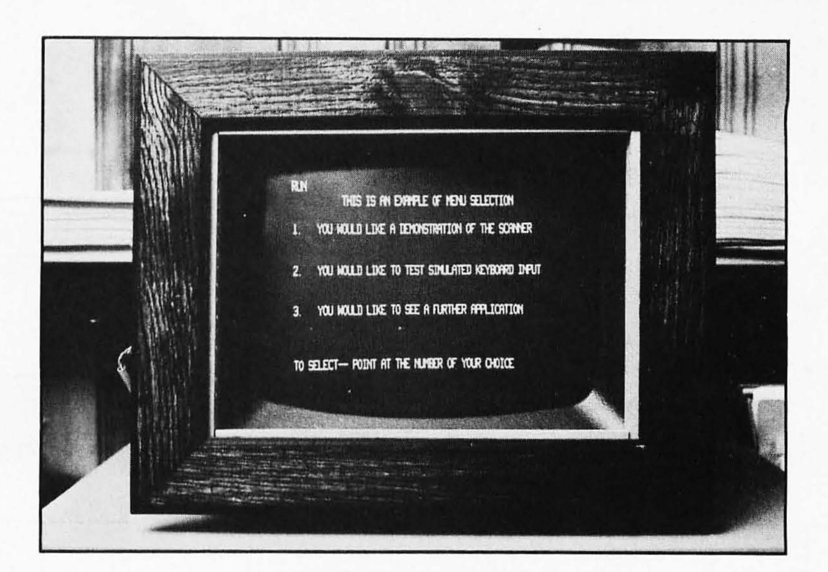


Photo 1a: Noncontact scanning digitizer in action with a BASIC program.

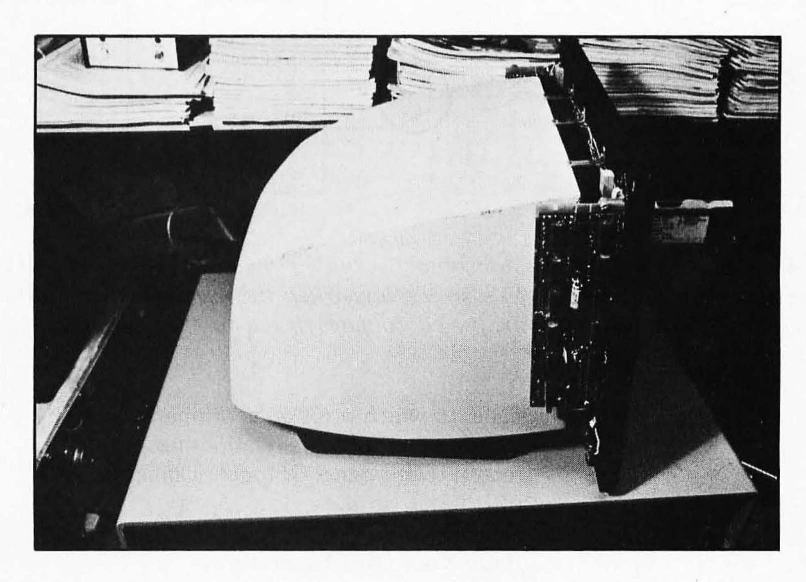


Photo 1b: Rear view of the video monitor showing circuitry mounted on two printed circuit boards on either side of the picture frame.

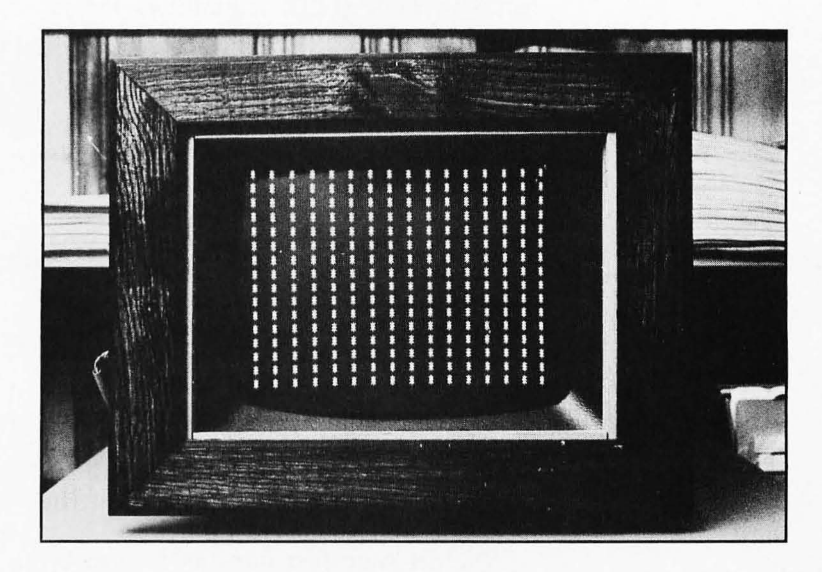


Photo 2: Display showing locations of the 256 points of the array.

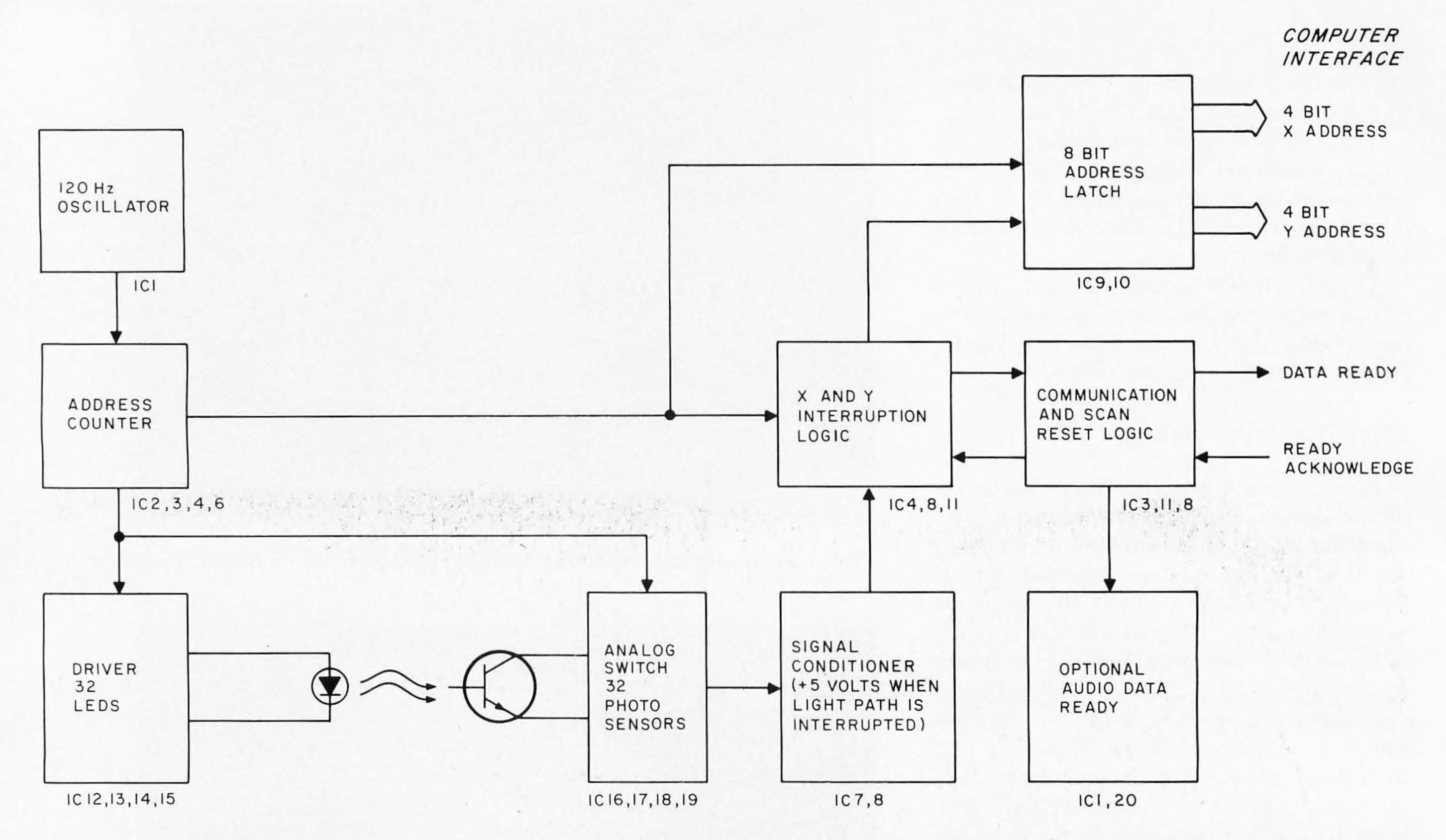


Figure 1: Block diagram of the noncontact scanning digitizer. Two rows of 16 pairs of LEDs and phototransistors are placed opposite each other in front of a video display. When the user breaks the infrared light beams with a finger or other object, a signal is sent to the computer giving the coordinates of the point in question.

paper in which position coordinates could be translated into usable relationships. I refer to it as a touch panel or touch scanner for lack of a better word.

Build Your Own Touch Panel

The touch panel is an elaborate infrared scanner. There are 32 pairs of infrared light emitting diode (LED) transmitters and receivers mounted around the perimeter of the screen. There are 16 on the X (or horizontal) axis and 16 on the Y (or vertical) axis. The resolution of such a device is therefore 16 by 16, and there are 256 individual points. Photo 2 shows this grid system.

Figure 1 is the block diagram of the system, and figure 2 shows the detailed schematics of the system. The noncontact digitizer is basically a hardware stepping circuit that turns on each transmitter/receiver pair sequentially and checks to see if anything (like a finger or a pencil) is blocking the beam. The transmitters and receivers are on opposite sides of the board, as illustrated in figure 3. The lower left corner is position (0,0) in a Cartesian coordinate system. The upper right is location (15,15).

The hardware first turns on the pair D_0 and Q_0 and then sequences down the line along the horizontal (X) axis to D_{15} and Q_{15}. Only one pair is energized at any one time. If any of the beams within these 16 pairs is obstructed, the 4 bit binary code for that location is loaded into IC9. The scan continues in the Y direction in a similar manner and the 4 bit Y position is loaded into IC10. If the hardware senses that something is obstructing an X and Y beam within one scan around the perimeter, it sets a data ready flag and stops the scanner.

The data presented to the computer is an 8 bit word representing a 4 bit X coordinate and a 4 bit Y coordinate. These lines are simply tied to a parallel input port, in the same manner as all the other devices I design. The data ready bit can be read either as a single bit input on another port, or as a control line on a more intelligent interface. When the program senses that the data ready is high, it reads the scanner data and momentarily pulses the ready reset line low to start the scan cycle again.

Use a Picture Frame

The heart of the system is the LEDs and phototransistors shown in photo 3. The device on the left is a General Electric LED 56 and the photodarlington detector used with

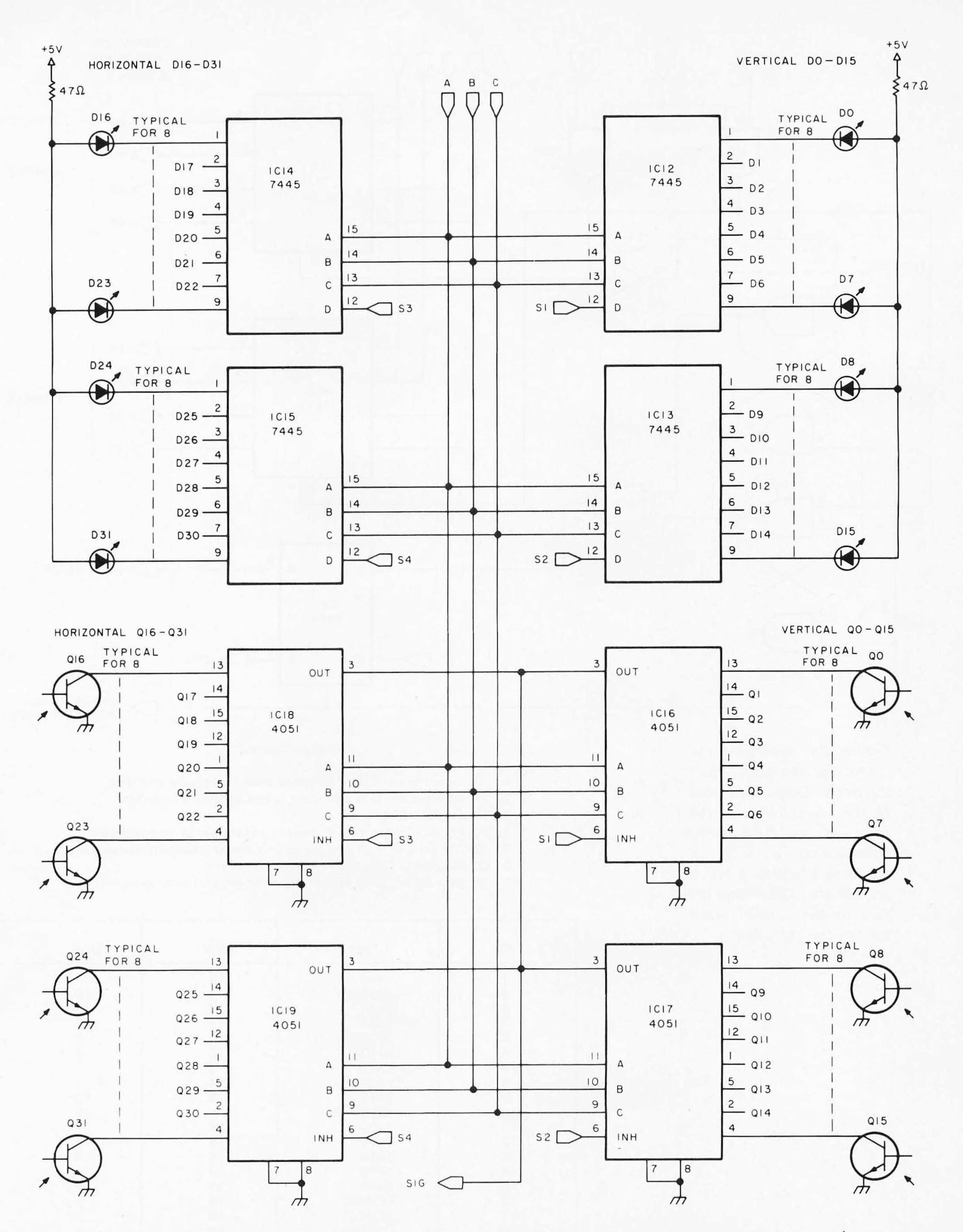


Figure 2a: LED driver and optical receiver circuitry for the noncontact digitizer. Each transmitter/receiver pair (consisting of an LED and phototransistor) is activated sequentially via lines A, B and C. D0 and Q0 are turned on first, and the sequence continues down the horizontal axis to D15 and Q15. If any of the beams is broken, the 4 bit binary code for that location is loaded into IC9 (see figure 2b). The scan continues in the Y direction and the 4 bit Y position is loaded into IC10. Any obstruction causes the data ready flag to be set and the scanner to be halted.

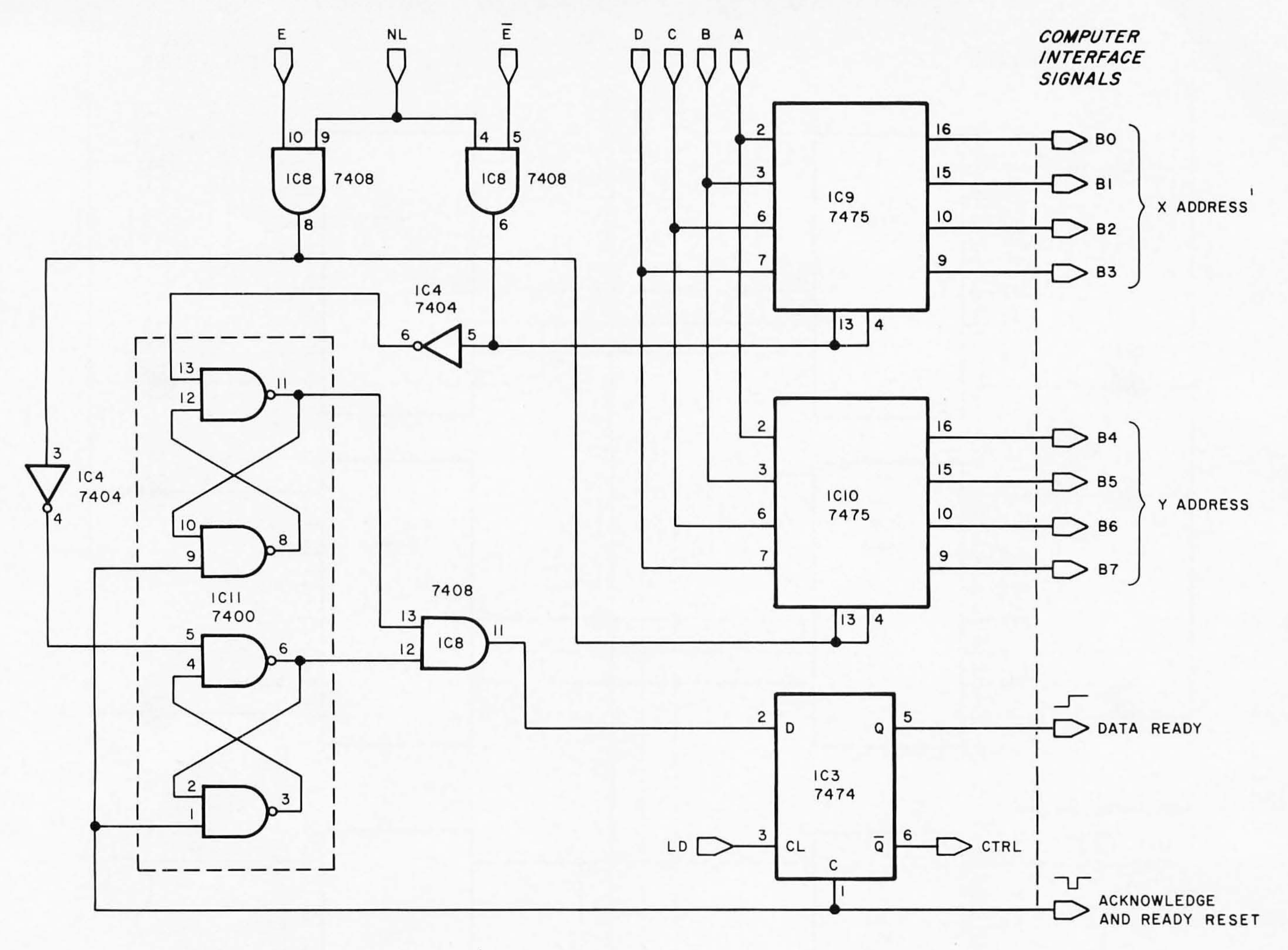


Figure 2b: Interface circuitry for the noncontact digitizer. Data presented to the computer is in the form of an 8 bit word representing a 4 bit X coordinate and a 4 bit Y coordinate. These lines are tied to the parallel input port of the computer.

Notes on figure 2

1. All capacitors are 25 V ceramics unless otherwise specified.
2. All resistors are ¼ W 5 percent unless otherwise specified.
3. ⏚ denotes signal ground.
4. ICs 16 thru 19 are CMOS devices and should be handled carefully.
5. Additional LEDs on prototype unit are for testing purposes only.
6. Q0 thru Q31: GE LED56 infrared emitter.
   D0 thru D31: GE L14F2 photodarlington infrared detector.

IC	Type	+5 V	Gnd
1	7400	14	7
2	7493	5	10
3	7474	14	7
4	7404	14	7
5	74155	16	8
6	74123	16	8
7	LM311	8	1
8	7408	14	7
9	7475	5	12
10	7475	5	12
11	7400	14	7
12	7445	16	8
13	7445	16	8
14	7445	16	8
15	7445	16	8
16	CD4051	16	8
17	CD4051	16	8
18	CD4051	16	8
19	CD4051	16	8
20	74121	14	7

Table 1: Power wiring table for the noncontact digitizer.

Figure 2c: Address decoder and phototransistor signal conditioning circuitry for the noncontact digitizer. IC2 is a counter driven by the oscillator at upper left. When a phototransistor is activated, the SIG line goes high, activating line NL, which stores the 4 bit address of the interrupted beam (see figure 2b). The scanner is finally halted via the CTRL line. The computer then reads the coordinates and reactivates the scanner.

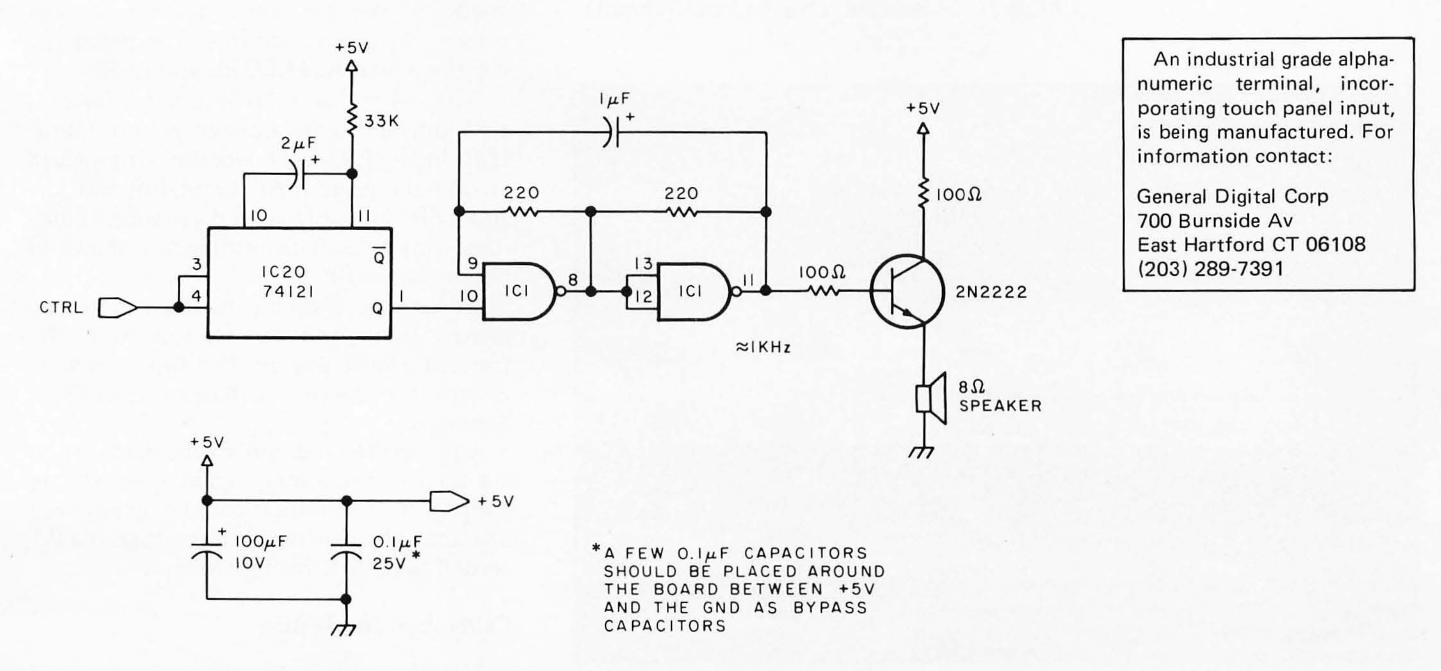


Figure 2d: Optional audio data ready signal circuit, which causes an audible beep on a speaker whenever a pair of beams is obstructed and sets the data ready signal.

An industrial grade alphanumeric terminal, incorporating touch panel input, is being manufactured. For information contact:

General Digital Corp
700 Burnside Av
East Hartford CT 06108
(203) 289-7391

Photo 3: GE LED56 light emitting diode (left) and GE L14F2 photodarlington infrared detector used to transmit and receive infrared light, respectively, for use in the noncontact scanning digitizer.

it is the L14FZ. These units have built-in glass lenses and are very sensitive. A much less expensive though equally capable opto-electric pair is the H17B1 shown on the right in photo 3. Because it has no lens, it requires considerably more shielding from ambient light, but it will work if properly aligned. I have checked the operation of both devices and recommend the lensed type if you intend to use the touch scanner in high ambient light environments. The prototype described here used LED56s and L14Fs.

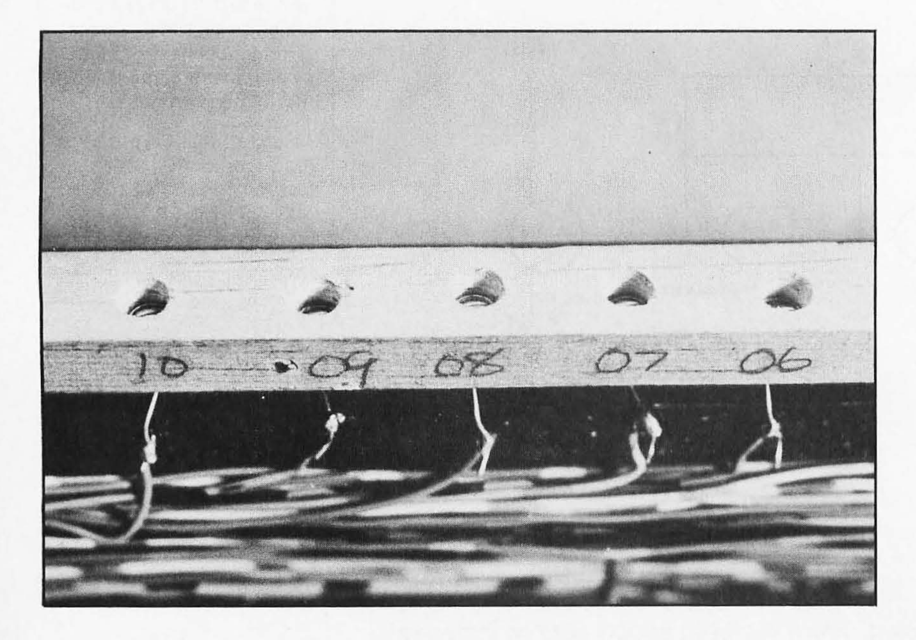


Photo 4a: Mounting the photodarlington detectors.

The frame that holds all the electronics is a $4 discount store wooden picture frame. Half inch (1.27 cm) wooden strips glued around the edges hold the phototransistors and LEDs in evenly spaced, recessed, ¼ inch (0.63 cm) holes. This technique is shown in photos 4a and 4b.

The entire assembly is attached to the picture frame and can be secured to the front of a video display. The display in these photos is a 12 inch (30.76 cm) surplus Phase 4 monitor.

One further addition to the hardware to aid users of the scanner is audio feedback to confirm that a position coordinate has been selected. The data ready strobe triggers a 0.1 second beep on a small speaker.

Calibration and Testing

There is virtually nothing to calibrate or test on this unit. The only adjustment is the

Notes for figure 3

1. Scan is sequential from D0 thru D31.
2. Only one LED is on at any one time.
3. Scan rate is approximately four samples per second.
4. Total detectable points = 256.

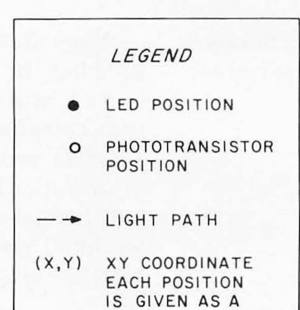
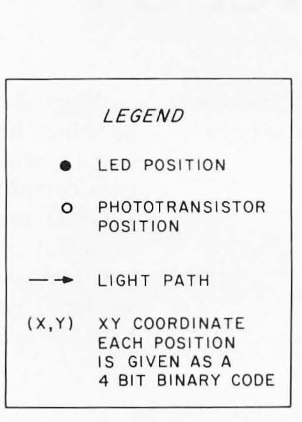
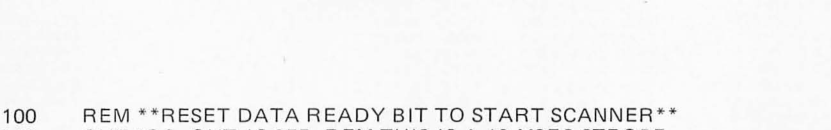



Figure 3: LED and phototransistor placement for the 16 by 16 Cartesian coordinate noncontact digitizer.

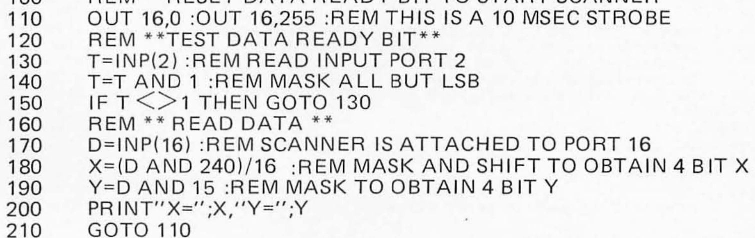


sensitivity control on the phototransistor amplifier. Direct sunlight or incandescent lights will cause saturation of the input and disable the scanner. The only other important consideration is mechanical alignment: the LED and phototransistor constituting each pair must be exactly opposite and in direct alignment.

The program in listing 1 is a simple BASIC program to exercise the scanner and provide the operator with an indication of its operational integrity. It is written in Micro Com 8 K Zapple BASIC. The decimal coordinates of X and Y will be output as your finger is moved across the scanned area. This is the only routine that has to be added to any BASIC program to exercise the scanner. If set up as a subroutine by changing line 210 to a RETURN statement, the routine will turn the scanner on when called and return to the main program with a value in variable D representing the coordinates to which you pointed. The main program then responds appropriately.

Obviously the scanner would be more efficiently driven by a machine language program, but I feel most users will be interested in utilizing this device with a high level language. The relatively slow scan rate allows considerable leeway.

Next month I'll pursue the software (in BASIC) necessary to drive this scanner effectively. The major emphasis will be the use of menus and keyboard substitutions.■



```
100   REM **RESET DATA READY BIT TO START SCANNER**
110   OUT 16,0 :OUT 16,255 :REM THIS IS A 10 MSEC STROBE
120   REM **TEST DATA READY BIT**
130   T=INP(2) :REM READ INPUT PORT 2
140   T=T AND 1 :REM MASK ALL BUT LSB
150   IF T<>1 THEN GOTO 130
160   REM ** READ DATA **
170   D=INP(16) :REM SCANNER IS ATTACHED TO PORT 16
180   X=(D AND 240)/16  :REM MASK AND SHIFT TO OBTAIN 4 BIT X
190   Y=D AND 15 :REM MASK TO OBTAIN 4 BIT Y
200   PRINT"X=";X,"Y=";Y
210   GOTO 110
```

Listing 1: Program written in 8 K Zapple BASIC to exercise the scanner.

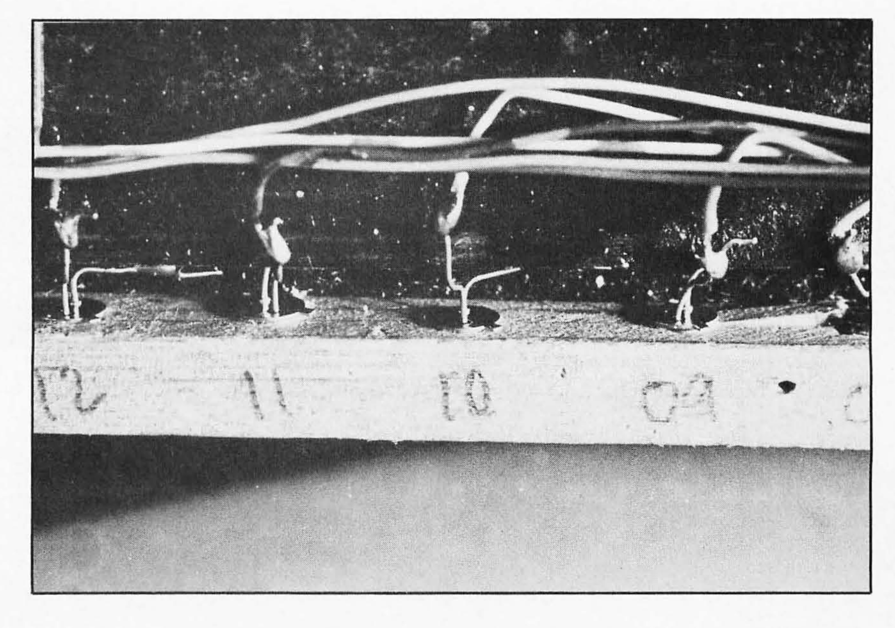


Photo 4b: Mounting the LEDs.

REFERENCES

1. Loomis, Sumner S, "Let There Be Light Pens," January 1976 BYTE, page 26.
2. Webster and Young, "Add a $3 Light Pen to Your Video Display," February 1978 BYTE, page 52.

JACPOT

Edwin E Hastings
18 Churchill Rd
Marblehead MA 01945

Many states are now working to legalize gambling, but why wait for the bureaucratic process when you can start a casino on your own computer (for fun of course) with the JACPOT program. JACPOT is a simulation of a slot machine written in BASIC, with bets limited only by the constraints of your machine. You can lose all your money and then lose it again!

Listing 1: The JACPOT game, written in BASIC. [We detect a small amount of bias in the symbols for a winning combination, which can of course be redefined arbitrarily to suit the user's fancy. . .**CH**]

```
JACPOT  10-MAY-77  MU BASIC/RT-11 V01-01

10 REM COPYRIGHT 1977 BY EDWIN E. HASTINGS
20 RANDOMIZE
30 PRINT "    PLEASE ENTER THE AMOUNT OF MONEY"
40 PRINT "    YOU WISH TO PLAY WITH ";
50 INPUT Z
60 IF Z<0 THEN 80
70 GO TO 90
80 PRINT "    PLEASE ENTER A POSITIVE NUMBER." \ GO TO 50
90 PRINT
100 PRINT "    DO YOU WANT INSTRUCTIONS(Y-N)";
110 LET A$=CHR$(SYS(4))
120 IF A$="N" THEN 310 \ IF A$<>"Y" THEN 100
130 PRINT
140 PRINT "    BANDIT---A SIMULATED SLOT MACHINE"
150 PRINT "    PLAY UNTIL YOU OR THE BANK ARE BROKE"
160 PRINT "    WINNING COMBINATIONS ARE AS FOLLOWS."
170 PRINT
180 PRINT "        8080 8080 ????    PAYS   2:1"
190 PRINT "        8080 IBM  ????    PAYS   2:1"
200 PRINT "        8080 8080 IBM     PAYS   4:1"
210 PRINT "        8080 8080 8080    PAYS   4:1"
220 PRINT "        6800 6800 6800    PAYS   8:1"
230 PRINT "        6502 6502 6502    PAYS  16:1"
240 PRINT "        Z-80 Z-80 Z-80    PAYS  32:1"
250 PRINT "        CARD CARD CARD    PAYS  64:1"
260 PRINT "        TAPE TAPE TAPE    PAYS 128:1"
270 PRINT "        DISK DISK DISK    PAYS 256:1"
280 PRINT "        IBM  IBM  IBM       SUPRISE"
290 PRINT "        DEC  DEC  DEC     TAKES IT ALL"
300 PRINT
310 PRINT "    WHEN A QUESTION MARK IS PRINTED,RESPOND WITH THE"
320 PRINT "    NUMBER OF REPEATED PLAYS YOU WISH TO MAKE,"
330 PRINT "    A COMMA,AND THE SIZE OF YOUR BET.IF YOU WISH"
340 PRINT "    TO STOP BEFORE YOU BREAK THE BANK,OR YOU GO BROKE,"
350 PRINT "    INPUT A '0,0' WHEN YOUR BETS ARE ASKED FOR."
360 PRINT  \ PRINT
370 LET B1=INT(1.85*Z) \ LET P1=Z
380 LET C1=0 \ LET C2=0
390 IF C2<C1 THEN 510
400 IF B1<=0 THEN 1480
410 GO TO 480
420 PRINT "    YOUR BETS PLEASE"; \ INPUT C1,C4
430 PRINT
440 IF C1=0 THEN IF C4=0 THEN 1670
450 IF C4<=0 THEN 420
460 LET C2=0
470 GO TO 510
480 PRINT "    THE BANDIT HAS $"B1",YOU HAVE $"P1
490 PRINT
500 GO TO 420
510 LET P1=P1-C4 \ LET B1=B1+C4
520 LET C2=C2+1
530 GOSUB 740
540 LET A1=X2
550 GOSUB 740
560 LET A2=X2
570 GOSUB 740
580 LET A3=X2
590 IF A1<>9 THEN 600 \ LET A1=A2
600 IF A2<>9 THEN 610 \ LET A2=A1
610 IF A3<>9 THEN 620 \ LET A3=A2
620 IF (A1*A2)<>81 THEN 640
630 LET A2=A3 \ LET A1=A2
640 IF A1=A2 THEN 660
650 GO TO 1180
660 IF A2=A3 THEN 720 \ IF A1=2 THEN 680
670 GO TO 1180
680 IF A3=1 THEN 1180
690 LET B1=B1-(2*C4) \ LET P1=P1+(2*C4)
700 PRINT "WIN   $";2*C4
710 GO TO 390
720 IF A1=1 THEN 1180
730 GO TO 1370
740 LET X2=0
750 LET X1=INT(10000*RND(1))
760 IF X1>7578 THEN 750
770 IF X1>7514 THEN 870
780 IF X1>7412 THEN 880
790 IF X1>7248 THEN 890
800 IF X1>6968 THEN 900
810 IF X1>6566 THEN 910
820 IF X1>5894 THEN 920
830 IF X1>4818 THEN 930
840 IF X1>3096 THEN 940
850 IF X1>340 THEN 950
860 GO TO 960
870 LET X2=X2+1
880 LET X2=X2+1
890 LET X2=X2+1
900 LET X2=X2+1
910 LET X2=X2+1
920 LET X2=X2+1
930 LET X2=X2+1
940 LET X2=X2+1
950 LET X2=X2+1
960 LET X2=X2+1
970 IF X2<10 THEN 990
980 PRINT "DEC   "; \ RETURN
990 IF X2<9 THEN 1010
1000 PRINT "IBM   "; \ RETURN
1010 IF X2<8 THEN 1030
1020 PRINT "DISK  "; \ RETURN
1030 IF X2<7 THEN 1050
1040 PRINT "TAPE  "; \ RETURN
1050 IF X2<6 THEN 1070
1060 PRINT "CARD  "; \ RETURN
1070 IF X2<5 THEN 1090
1080 PRINT "Z-80  "; \ RETURN
1090 IF X2<4 THEN 1110
1100 PRINT "6502  "; \ RETURN
1110 IF X2<3 THEN 1130
1120 PRINT "6800  "; \ RETURN
1130 IF X2<2 THEN 1150
1140 PRINT "8080  ";
1150 IF X2>1 THEN 1170
1160 PRINT "WANG  ";
1170 RETURN
1180 LET A4=0
1190 IF A1=10 THEN 1260 \ IF A2=10 THEN 1260 \ IF A3=10 THEN 1260
1200 IF A1>1 THEN 1210 \ LET A4=A4+1
1210 IF A2>1 THEN 1220 \ LET A4=A4+1
1220 IF A2>1 THEN 1230 \ LET A4=A4+1
1230 IF A4>0 THEN 1290
1240 PRINT
1250 GO TO 1580
1260 PRINT "WIN   $"10*C4" WITH DEC"
1270 LET B1=B1-(10*C4) \ LET P1=P1+(10*C4)
1280 GO TO 1580
1290 IF A4>1 THEN 1320
1300 LET B1=B1+(2*C4) \ LET P1=P1-(2*C4)
1310 PRINT "ZILCH-YOU LOSE $"2*C4 \ GO TO 1580
1320 IF A4>2 THEN 1350
1330 LET B1=B1+(4*C4) \ LET P1=P1-(4*C4)
1340 PRINT "DOUBLE ZILCH-YOU LOSE $"4*C4 \ GO TO 1580
1350 LET B1=B1+(8*C4) \ LET P1=P1-(8*C4)
1360 PRINT "TRIPLE ZILCH-YOU LOSE $"8*C4 \ GO TO 1580
1370 LET X4=2
1380 FOR X3=2 TO 10
1390 LET X4=X4*2
1400 IF A1=X3 THEN 1420
1410 NEXT X3
1420 IF A1=10 THEN 1460 \ IF A1=9 THEN 460
1430 LET B1=B1-(X4*C4) \ LET P1=P1+(X4*C4)
1440 PRINT "WIN   $"X4*C4
1450 GO TO 1580
1460 LET P1=P1*2
1470 LET B1=0
1480 PRINT
1490 PRINT TAB(15);"JACKPOT";CHR$(7);CHR$(7)
1500 PRINT "    YOU BROKE THE BANK,YOU NOW HAVE $"P1","
1510 PRINT "    PLEASE LEAVE QUIETLY"
1520 GO TO 1630
1530 LET Z2=INT(2/5*Z)
1540 LET P1=P1+Z2 \ LET B1=B1-Z2
1550 PRINT "IT'S MAGIC BONUS TIME,YOU WIN "Z2" DOLLARS."
1560 PRINT "COMPLIMENTS OF DEC!";CHR$(7)
1570 FOR K=1 TO 4 \ PRINT  \ NEXT K
1580 IF B1<=0 THEN 1490
1590 IF P1<=0 THEN 1620
1600 GO TO 390
1610 PRINT
1620 PRINT "    YOU'RE ALL DONE,NO CREDIT ALLOWED."
1630 FOR K2=1 TO 4 \ PRINT  \ NEXT K2
1640 PRINT "DO YOU WANT TO PLAY AGAIN(Y-N)";
1650 LET Y$=CHR$(SYS(4))
1660 IF Y$="Y" THEN 310 \ IF Y$<>"N" THEN 1650
1670 END

READY
```

Listing 2: A sample run of JACPOT.

```
JACPOT  10-MAY-77  MU BASIC/RT-11 V01-01

    PLEASE ENTER THE AMOUNT OF MONEY
    YOU WISH TO PLAY WITH ? 1000

    DO YOU WANT INSTRUCTIONS(Y-N)Y
    BANDIT---A SIMULATED SLOT MACHINE
    PLAY UNTIL YOU OR THE BANK ARE BROKE
    WINNING COMBINATIONS ARE AS FOLLOWS.

        8080 8080 ????    PAYS   2:1
        8080 IBM  ????    PAYS   2:1
        8080 8080 IBM     PAYS   4:1
        8080 8080 8080    PAYS   4:1
        6800 6800 6800    PAYS   8:1
        6502 6502 6502    PAYS  16:1
        Z-80 Z-80 Z-80    PAYS  32:1
        CARD CARD CARD    PAYS  64:1
        TAPE TAPE TAPE    PAYS 128:1
        DISK DISK DISK    PAYS 256:1
        IBM  IBM  IBM      SUPRISE
        DEC  DEC  DEC     TAKES IT ALL

    WHEN A QUESTION MARK IS PRINTED,RESPOND WITH THE
    NUMBER OF REPEATED PLAYS YOU WISH TO MAKE,
    A COMMA,AND THE SIZE OF YOUR BET.IF YOU WISH
    TO STOP BEFORE YOU BREAK THE BANK,OR YOU GO BROKE,
    INPUT A '0,0' WHEN YOUR BETS ARE ASKED FOR.


    THE BANDIT HAS $ 1849 ,YOU HAVE $ 1000

    YOUR BETS PLEASE? 10,10

TAPE  6502  WANG
8080  8080  Z-80   WIN    $ 20
8080  6502  8080
6502  6800  6800
CARD  8080  8080
CARD  8080  8080
8080  CARD  6502
8080  8080  6800   WIN    $ 20
TAPE  8080  6800
6502  Z-80  TAPE
    THE BANDIT HAS $ 1909 ,YOU HAVE $ 940

    YOUR BETS PLEASE? 10,20

6800  6800  6800   WIN    $ 160
6502  6800  6800
6800  8080  TAPE
Z-80  8080  8080
Z-80  8080  6502
6502  8080  8080
6502  6502  6800
DEC   6502  CARD   WIN    $ 200   WITH DEC
TAPE  8080  Z-80
8080  8080  8080   WIN    $ 80
    THE BANDIT HAS $ 1669 ,YOU HAVE $ 1180

    YOUR BETS PLEASE? 10,100

Z-80  TAPE  8080
8080  6800  Z-80
6502  6502  8080
8080  WANG  6800   DOUBLE ZILCH-YOU LOSE $ 400
8080  8080  8080   WIN    $ 400
8080  6800  6502
CARD  6502  8080
6502  WANG  IBM    DOUBLE ZILCH-YOU LOSE $ 400
    YOU'RE ALL DONE,NO CREDIT ALLOWED.



DO YOU WANT TO PLAY AGAIN(Y-N)Y
    WHEN A QUESTION MARK IS PRINTED,RESPOND WITH THE
    NUMBER OF REPEATED PLAYS YOU WISH TO MAKE,
    A COMMA,AND THE SIZE OF YOUR BET.IF YOU WISH
    TO STOP BEFORE YOU BREAK THE BANK,OR YOU GO BROKE,
    INPUT A '0,0' WHEN YOUR BETS ARE ASKED FOR.


    THE BANDIT HAS $ 1849 ,YOU HAVE $ 1000

    YOUR BETS PLEASE? 11,5

WANG  6800  6502   ZILCH-YOU LOSE $ 10
8080  8080  8080   WIN    $ 20
8080  6502  8080
TAPE  8080  WANG
8080  8080  6800   WIN    $ 10
DEC   8080  8080   WIN    $ 50   WITH DEC
6800  8080  8080
8080  6800  DEC    WIN    $ 50   WITH DEC
8080  8080  WANG
8080  6502  CARD
TAPE  8080  8080
    THE BANDIT HAS $ 1784 ,YOU HAVE $ 1065

    YOUR BETS PLEASE? 10,10

WANG  6800  IBM    ZILCH-YOU LOSE $ 20
CARD  6800  8080
8080  8080  IBM    WIN    $ 40
Z-80  8080  6800
CARD  8080  6800
8080  6800  IBM
8080  8080  8080   WIN    $ 40
8080  8080  6800   WIN    $ 20
8080  8080  8080   WIN    $ 40
6502  8080  8080
    THE BANDIT HAS $ 1764 ,YOU HAVE $ 1085

    YOUR BETS PLEASE? 0,0


READY
```

The main object of JACPOT is to play until you or the house go broke. The winning combinations pay off in ratios of from 2:1 to 256:1 and more. In more than 1600 bets my payoff ratio has varied from 2:1 to 64:1.

In JACPOT there are different ways to win and lose. The first way to win is to hit a winning combination. The second is to hit a DEC in a combination. DEC pays off 10:1.

Table 1: A list of the major variables used in JACPOT and their functions.

Variable	Function
B1	The house's money.
C1	The number of bets.
C4	The size of the bets.
P1	The player's money (during the game).
X1	Determines the combinations.
X2	Prints the combinations.
Z	The player's money (start of game).

The way to lose money is to hit a ZILCH, which causes you to lose up to 8:1.

The program (see listing 1) was written on a Digital Equipment Corporation PDP-11, but can be easily modified for most BASIC interpreters. In lines 110 and 1650, the statement LET. . .=CHR$(SYS(4)) gives an automatic carriage return after the variable is inputted; this line can be replaced with an INPUT statement. Lines 870 to 960 look repetitive but they serve to determine the combinations. The backslashes in some lines are used to separate two or more lines. They are replaced by colons in some BASIC packages. In lines 1490 and 1560 the CHR$ (7) is used to ring the bell. The RANDOMIZE statement in line 20 is used to activate the random number generator. In some BASICs only the RND in line 750 is needed.

I hope that you have a good time running this program. If you have any questions or comments about this program, please write to me.■

Pascal versus BASIC: An Exercise

Allan M Schwartz
114-2 Nimitz Dr
West Lafayette IN 47906

Introduction

Pascal is one of the newest high level languages on the personal computing scene. Pascal has been accepted at many universities for several years. It is being used more and more in industry outside of education, and has just recently been introduced in microcomputers. Why is there so much enthusiasm about Pascal?

Pascal is a general purpose language, the product of the long evolution of computer languages. It has a simple but elegant syntax and has been implemented in both large systems (CDC 6000, IBM 360 and 370, Burroughs 6700, etc) and microcomputers (LSI-11, 8080, 8085 and Z-80).

Historical Background

Just as computer hardware has been continuously evolving during the past 25 years, so too have computer software requirements. Originally, computers were employed to work on mathematical tasks such as solving ballistics problems, or generating tables of logarithms. Later it became economically feasible to use computers for data processing or working with voluminous amounts of data such as census data or bank statements. Recently we have seen computers participate in various customized, dedicated applications like the control of traffic lights, microwave ovens and automobile ignitions.

We have seen a variety of applications and language requirements lead to an evolution of computer languages. "Programming" originally entailed the translation of simple algorithms into machine code and bit by bit loading of the computer's memory via the front panel. Later, assembly languages were used, followed by equation or formula translators such as FORTRAN. When it was discovered that computing involved mostly computing decisions and repetition, the language ALGOL (*ALGO*rithmic *L*anguage) was designed to express algorithms more clearly and conveniently. The need for a language to structure and represent all of the data and files in business data processing applications was filled by COBOL. Today we have Pascal, which has flexible data representations, sufficient flow of control statements to represent algorithms, and a clear, simple syntax making it a favorite for a variety of applications. Pascal is the result of several evolutionary steps in the history of computer languages.

Why is Pascal so appealing? First, it is an expressive language. It has several control structures that make the coding of algorithms very natural. Second, Pascal has flexible data representation.

Expression of Algorithms in Pascal

Figure 1 presents an algorithm to compute the greatest common divisor (GCD) of X and Y. The greatest common divisor of the integers X and Y is the largest integer that will divide evenly into both X and Y. Note that three assertions are stated in the flowchart. The first, a necessary precondition, states that X and Y must be positive integers. The second is a loop invariant such that, when control passes through that path in the flowchart, the GCD(X, Y) is equal to the GCD(A, B). The

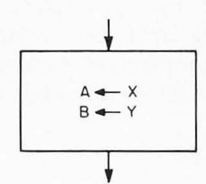
About the Author

Allan Schwartz is currently studying computer science at Purdue University and researching the applications of microcomputers to programming methodology. Previously, Mr Schwartz worked for Nielsen Engineering and Research Inc at NASA Ames Research Center as a systems programmer. He owns an MC6800 based microcomputer system and is developing a Pascal P-machine emulator for the 6800 microprocessor.

third, a post condition, states that A is equal to B, which is equal to the result, the GCD(X, Y).

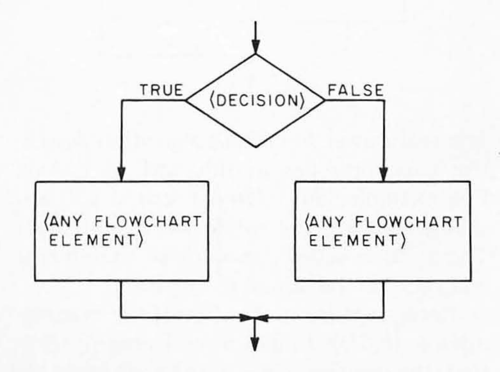
If we can prove these three points are true, then the algorithm is correct – that is, it will compute the greatest common divisor of X and Y. The loop invariance is easily proved, because if B is greater than A, the GCD(A,B) equals GCD(A, B–A) (a more rigorous proof is posed as an exercise in Wirth's book (see bibliography)). The post condition is also easy to prove, because the path to this exit is taken only when A equals B, and then the GCD(A, A) certainly equals A.

We are now reassured that if the precondition is true, the algorithm will compute the desired result. Now, how do we code this algorithm into our favorite programming language? Before we answer that question, let's look at the elements of the flowchart. The flowchart in figure 1, and indeed any computable algorithm, is made up of three elements: *sequence, selection* and *repetition.* Sequences are represented in the flowchart by rectangular boxes such as:



Note that this flowchart element has one entry (the arrow going in) and one exit. [*In BYTE's use of flowcharts, a top to bottom flow of control is assumed with arrows used for exceptions; in this article we make a stylistic exception, using extra arrows to emphasize flow. . .*CH]

The second flowchart element is selection. Selection is represented by:



A selection flowchart element requires at least two or three boxes; however, it always has one entry and one exit.

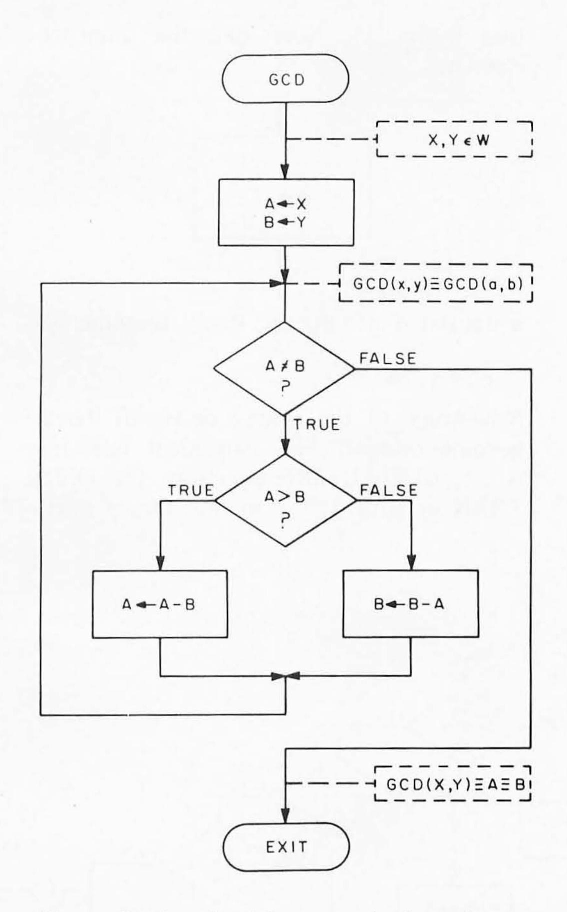
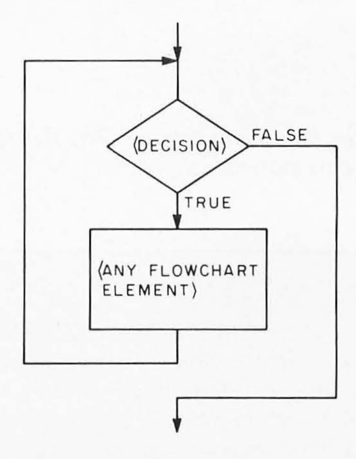


Figure 1: An algorithm to calculate the greatest common divisor (GCD) of two integers. (The greatest common divisor of two integers is the largest integer that will divide evenly into the two integers.)

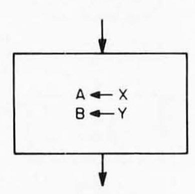
The third flowchart element is repetition. It is represented by:



This form of repetition is called a "while loop," because while the decision is true, the element is repeated. Again, this element has one entry and one exit.

These flowchart elements have been translated directly into Pascal statements

(see listing 1). Note that the sequence element:



is translated into the two Pascal assignments.

a := *x*; *b* := *y*

Now some of the syntax details of Pascal become evident. The assignment operator is :=, which is different from the FORTRAN or BASIC "=" in that the := operator in Pascal is used for assignment only, while the = in BASIC and FORTRAN is used as both the assignment operator and the equals sign. Statements are separated by semicolons, and any number of statements may be typed on one line. If the above sequence were a subelement of a selection element, it would be bracketed by **begin** and **end** keywords. For example:

```
if (x>0) and (y>0) then
   begin a := x; b := y
   end
```

Any number of elements combined into one sequence element by **begin** and **end** brackets forms a *compound statement.*

The selection flowchart element is translated into the Pascal **if** statement:

```
if a>b then a := a–b
        else b := b–a
```

And the repetition flowchart element is translated into the Pascal **while** statement:

while *a* <> *b* **do** <*statement*>

The expression <*statement*> is called a *metavariable*. For an explanation, see the accompanying text box. Notice, though, that the metavariable < *statement* > in the greatest common divisor **while** clause is an **if** statement.

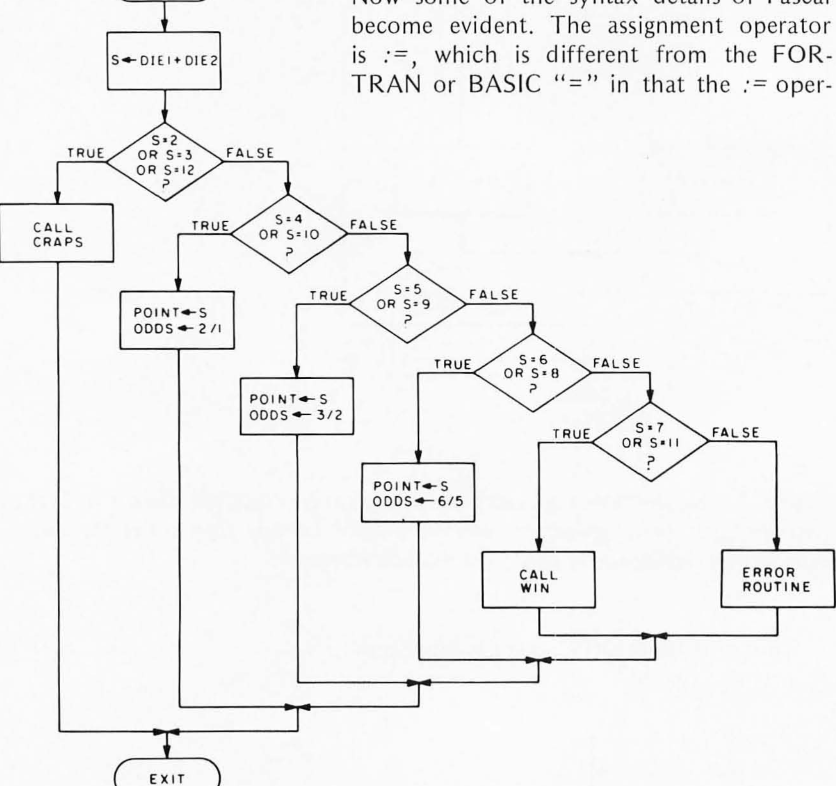
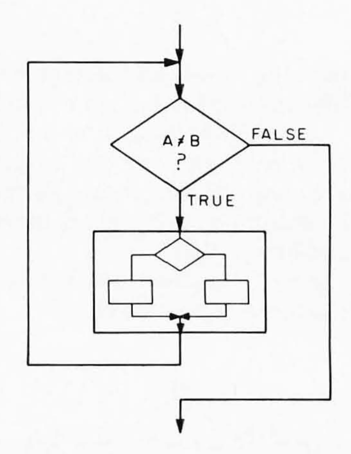


Figure 2: Flowchart for a portion of the dice game "craps." The five IF tests can be implemented in Pascal with one case statement.



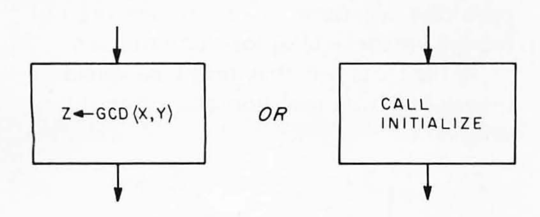
The real power in Pascal's algorithm descriptive capability lies in this sort of nesting. For example, any element can occur as a subelement of the **while** or **if** statement. These are called *structured* statements, and they can be nested to any depth.

Look again at the greatest common divisor (GCD) function in listing 1. Note that the routine consists of a heading and a variable declaration statement followed by one compound statement, bracketed by **begin** and **end**. Functions and proce-

Metavariables

Bracketed symbols such as ("< statement >") all call metalinguistic variables (or metavariables) or syntactic units. They represent a class of possible language elements. They are nonterminal symbols; that is, the symbol "< statement >" itself will not appear in a Pascal program. It represents a set of legal symbols that can appear in its place in the program. Nonterminal symbols are bracketed by "<" and ">" and are printed in italics to distinguish them from terminal symbols such as **for := if do**. *Terminal symbols are usually printed in heavy type if the symbol is a language key word, and appear exactly as they would in the Pascal program.*

dures in Pascal can be thought of as named statements with local variables. They always have one entry and one exit, and therefore, a call is flowcharted as a sequence element such as:



Pascal has a second selection statement called the **case** statement. This statement is a concise representation of the special case of nested **if** statements. An example of this is the "craps first roll" algorithm used to implement the dice game called craps. A pair of dice can obviously have only one summed value from 2 to 12 on any given throw, making this an ideal use for the case statement (see figure 2). The five nested decisions can be represented with the following Pascal **case** statement:

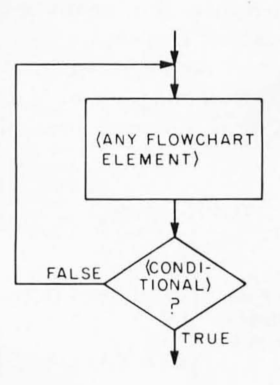
```
s : = die 1 + die 2;
case s of
   2, 3, 12 :
      craps;
   4, 10 :
      begin point : = s; odds : = 2/1
      end;
   5, 9 :
      begin point : = s; odds : = 3/2
      end;
   6, 8 :
      begin point : = s; odds : = 6/5
      end;
   7, 11 : win
end  {of case statement}
```

Of course, this could be represented using **if** statements; however, the **case** statement is much more concise and clear. When the decisions in a group of nested **if** statements are mutually exclusive, that is, if any one being true implies that the rest are false, then a **case** statement is probably the appropriate representation.

Pascal allows two other forms of repetition: the **repeat** statement and the **for** statement. The **repeat** statement:

```
repeat
   <any statement>
until <condition>
```

is represented by:



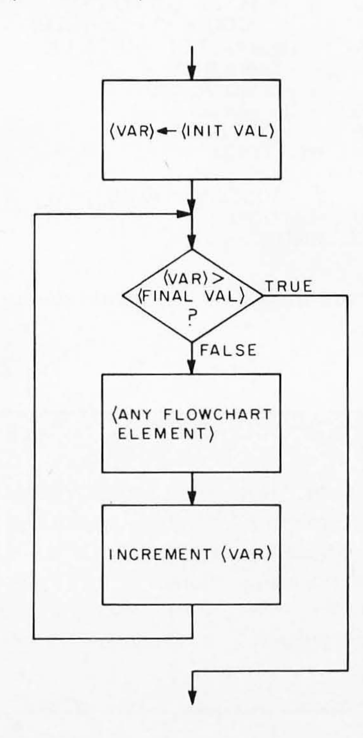
Repetitions can always be expressed as either **repeat** statements or **while** statements. However, one form usually sounds better. For example:

```
   repeat shoot craps
   until broke or out of time
is equivalent to
   shoot craps;
   while not broke
   and not out of time
   do shoot craps
```

The **for** statement:

```
for <var>:=
   <init val> to <final val>
   <any statement>
```

is represented by:



Notice that again there is one entry and one exit for this flowchart element.

Another element we might see in a flowchart is an arrow coming out of a subelement, perhaps to a different page of the flowchart. This exit from the normal flow of execution is the only use of the Pascal **goto** statement. Indeed, very few Pascal procedures need **goto** statements to express the algorithm. Goto statements can fog the otherwise clear logic of a routine.

A final element that might be found in flowcharts is an assertion and commentary such as:

GCD (X,Y) ≡ A ≡ B

The Pascal greatest common divisor (GCD) function has all of these elements in an appropriate place in the source code. Pascal allows comments, delimited with braces, { and } , to be freely inserted anywhere a blank can be inserted.

We can conclude that for each Pascal language statement there is a corresponding flowchart element, and vice versa. Therefore, one could easily flowchart any algorithm just from its Pascal listing. Compare the Pascal program in listing 1 to the FORTRAN and BASIC programs in listings 2 and 3. They are fundamentally identical, but all of the statement numbers and GOTOs in the FORTRAN and BASIC versions obscure the logic. You might maintain that, for so simple an example, there is no advantage for Pascal. One could flowchart the greatest common divisor (GCD) algorithm just from the BASIC listing. Of course you could, but how about flowcharting that 1200 line FORTRAN headache you wrote a year ago that has returned to haunt you?

```
function gcd(x,y: integer): integer;
   var a,b: integer;          {x,y > 0}
   begin
      a := x;  b := y;
      while a <> b do     {GCD(X,Y) = GCD(A,B)}
         if a>b then a := a-b
                 else b := b-a;
                          {GCD(X,Y) = A = B}
      gcd := a
   end
```

Listing 1: Pascal function to calculate the greatest common divisor of two integers.

```
100  LET A=X
110  LET B=Y
120  IF A=B THEN 190
130     REM . . . GCD(X,Y) = GCD(A,B)
140     IF A>B THEN 170
150        LET B=B-A
160        GO TO 180
170        LET A=A-B
180  GO TO 120
190  REM . . . GCD(X,Y) = A = B
200  RETURN
```

Listing 2: BASIC subroutine to compute the greatest common divisor of two integers.

```
      INTEGER FUNCTION GCD(X,Y)
      INTEGER A,B,X,Y
      A=X
      B=Y
  120 IF (A.EQ.B)  GO TO 190
C        . . . GCD(X,Y) = GCD(A,B)
         IF (A.GT.B)  GO TO 170
           B = B-A
           GO TO 180
  170      A = A-B
  180    CONTINUE
      GO TO 120
C
C     . . . GCD(X,Y) = A = B
  190 RETURN
      END
```

Listing 3: FORTRAN function to compute the greatest common divisor of two integers.

Pascal's Namesake

Blaise Pascal (1623-1662), one of the foremost famous French mathematicians, developed the Pascal theorem of projective geometry at the age of 16. One year later he started developing a calculating machine. He completed the first operating model in 1642 and built 50 more during the next ten years. In 1654 he produced two papers establishing the foundations of integral calculus and of probability theory.

Data Representation in Pascal

Pascal has several flexible forms of data representation. A variable can be defined as a scalar (single value) or a structured type. The different scalar types are: real, integer, character, Boolean, and user defined or enumerated. The structured types include arrays, records, sets and files.

Users can define their own scalar types by enumeration. For example, in a traffic control program, there might be a variable called *signalcolor* which has a value of yellow, green or red. Or, in a microwave oven program, there might be a variable called *temp* which represents the cooking level specified. These concepts are represented by the following Pascal declarations:

type *color = (red,yellow,green);*
cooking level = (warm,defrost,simmer, roast,reheat, maxpower);
var *signalcolor: color;*
temp: cookinglevel;

In this example the **type** declaration describes the user defined types and the **var** declaration specifies variable names and their associated type.

Another innovation in Pascal is the ability to specify a subrange of a scalar type. For example, if the variable *count* is to be an integer between 1 and 10, the declaration would be:

var *count: 1. .10;*

To further demonstrate these features, a BASIC program that would benefit from Pascal data representation is next explored.

Mastermind Codebreaker Example

The Mastermind codebreaker algorithm I have chosen for this exercise was presented by WL Milligan in the October 1977 BYTE, pages 168 thru 171. His BASIC version is reproduced in listing 4. A Pascal translation is presented here in listing 5. Let us compare the two.

The first 15 lines of the Pascal version correspond to lines 10 to 45 in the BASIC version. These are the type declarations and the global variable declarations. These global variables can be referenced from within any procedure. The type declarations define new variable types such as:

type *colors* = *(colorless, red, blue, brown, green, yellow, orange, space);*
row = array *[1. .4]* of *colors;*
eval = record
black,white: 0. .4
end;

This means that a variable of type *colors* has a value equal to one of these enumerated items. A variable of type *row* is an array of four *colors*. The type *eval* represents a codemaker's response to a guessed row. What does this represent in the game? This response is the number of exact color and position matches (black key pegs) and the number of out of position color matches (white key pegs). The codemaker responds with between 0 and 4 black and white key pegs. The type *eval* in the Pascal version accurately models this: a record consisting

```
10 REM MASTER MIND 'CODEBREAKER'
20 REM CODED IN RT-11 BASIC
30 RANDOMIZE
40 DIM R$(9,3),S(9,1)
45 DIM A$(6),B$(3),C$(3),D$(3)
50 REM INITIALIZATION
60 FOR J=0 TO 6
70 READ A$(J)
80 NEXT J
90 DATA 'RED','BLUE','GREEN','YELLOW','BLACK','WHITE','SPACE'
100 LET L0=0
110 LET L1=0
120 LET L2=0
130 LET L3=0
140 PRINT 'MASTER MIND CODEBREAKER'
145 PRINT 'PLEASE BE PATIENT. SOMETIMES I TAKE A FEW MINUTES ON MY MOVE'
150 PRINT 'WHICH VERSION (1 OR 2) ';
160 INPUT V
170 LET V=V+5
180 REM ASSIGN COLORS AT RANDOM FOR ROW 1
190 FOR J=0 TO 3
200 LET R$(0,J)=A$(INT(V*RND(J)))
210 NEXT J
220 REM START MAIN PLAY OF GAME HERE
230 REM I IS THE ROW COUNTER
240 FOR I=0 TO 9
245 PRINT
250 PRINT 'MY MOVE FOR ROW'I+1' IS'
260 PRINT R$(I,0),R$(I,1),R$(I,2),R$(I,3)
270 PRINT 'HOW MANY BLACK PEGS  ';
280 INPUT S(I,0)
290 IF S(I,0)<>4 THEN 320
300 PRINT 'THANKS FOR THE GAME'
305 PRINT
310 GO TO 870
320 IF S(I,0)<>3 THEN 360
330 LET S(I,1)=0\REM IF 3 BLACKS THEN 0 WHITES
340 GO TO 380
360 PRINT 'HOW MANY WHITE PEGS  ';
370 INPUT S(I,1)
380 REM GENERATE HYPOTHESIS
390 FOR I0=L0 TO V-1
400 FOR I1=L1 TO V-1
410 FOR I2=L2 TO V-1
420 FOR I3=L3 TO V-1
430 LET D$(0)=A$(I0)
440 LET D$(1)=A$(I1)
450 LET D$(2)=A$(I2)
460 LET D$(3)=A$(I3)
470 REM CHECK ALL ROWS FROM FIRST TO CURRENT FOR CONSISTENCY
490 FOR R=0 TO I
500 FOR J=0 TO 3
510 LET C$(J)=R$(R,J)
520 LET B$(J)=D$(J)
530 NEXT J
540 REM USE ROW EVALUATION SUBROUTINE TO CHECK CONSISTENCY OF
550 REM HYPOTHESIS AGAINST EACH ROW
555 LET N=0\LET M=0
560 GOSUB 910
570 REM CHECK FOR AGREEMENT OF BLACK & WHITE COUNT
580 IF N<>S(R,0) THEN 700
590 IF M<>S(R,1) THEN 700
600 NEXT R
610 REM MAKE SURE THAT HYPOTHESIS ROW DOESNT DUPLICATE ROW 1
620 LET Z=0
630 FOR J=0 TO 3
640 IF R$(0,J)<>D$(J) THEN 660
650 LET Z=Z+1
660 NEXT J
670 IF Z=4 THEN 700
690 GO TO 820
700 NEXT I3
710 NEXT I2
720 NEXT I1
730 NEXT I0
740 PRINT 'I HAVE REACHED AN IMPASSE IN MY THINKING'
750 PRINT 'COULD YOU HAVE MADE AN ERROR?'
760 GO TO 870
770 LET L0=I0
780 LET L1=I1
790 LET L2=I2
800 LET L3=I3+1
810 REM DO NOT RECHECK ELIMINATED POSSIBILITIES
820 REM ASSIGN NEXT ROW
830 FOR J=0 TO 3
840 LET R$(I+1,J)=D$(J)
845 NEXT J
850 NEXT I
860 PRINT 'I AM STUMPED -- YOU WIN'
870 PRINT 'ANOTHER GAME  ';
880 INPUT R$
890 IF R$='Y' THEN 150
900 STOP
910 REM SUBROUTINE TO EVALUATE RESPONSE
920 REM COUNT BLACKS FIRST
930 FOR J1=0 TO 3
940 IF C$(J1)<>B$(J1) THEN 960
950 LET N=N+1
960 NEXT J1
970 REM NOW COUNT WHITES
980 FOR J1=0 TO 3
990 FOR J2=0 TO 3
1000 IF J1=J2 THEN 1080
1010 IF C$(J1)=B$(J1) THEN 1080
1020 IF C$(J2)=B$(J2) THEN 1080
1030 IF C$(J1)<>B$(J2) THEN 1080
1040 LET M=M+1
1050 LET B$(J2)='X'\REM DUMMY WRONG VALUE
1070 GO TO 1090
1080 NEXT J2
1090 NEXT J1
1100 RETURN
1110 STOP
2000 END
```

Listing 4: Codebreaker portion of W Lloyd Milligan's Mastermind game written in BASIC. The program appeared originally in the October 1977 BYTE, pages 169 and 170 (see page 176 of this issue for a description of Mastermind). Compare this with the Pascal version in listing 5.

Listing 5: Pascal version of the Mastermind BASIC program in listing 4.

```
program mm2(input,output);

label   870;

type    colors = (colorless, red, blue, brown, green, yellow, orange, space);
        row    = array [1..4] of colors;
        eval   = record
                    black, white: 0..4
                 end;

var     evaluations: array [1..10] of eval;
        rows:        array [1..10] of row;
        name:        array [colors] of packed array [1..6] of char
        color:       array [0..7] of colors;
        redrow: row;  { First hypothesis checked }
        last:   row;  { Last hypothesis formed   }
        version: 1..2;    maxcolor: orange..space;
        i: 1..9;  j: 1..4;  ch: char;


procedure initialization;
    var c: colors;  i: 1..4;
    begin                                    name[green]  := 'GREEN  ';
        name[red]      := 'RED    ';  name[yellow] := 'YELLOW';
        name[blue]     := 'BLUE   ';  name[orange] := 'ORANGE';
        name[brown]    := 'BROWN  ';  name[space]  := 'SPACE  ';
        for c := colorless to space do
            color[ord(c)] := c;
        for i := 1 to 4 do
            redrow[i] := red;
        last := redrow;
        writeln(' MASTERMIND CODEBREAKER');
        writeln(' PLEASE BE PATIENT. SOMETIMES I TAKE A FEW');
        writeln(' MINUTES ON MY MOVE.  WHICH VERSION (1 or 2)?');
        read (version);
        maxcolor := color[version+5];
        { Assign colors at random for row 1 }
        for i := 1 to 4 do
            rows[1,i] := color[ trunc((version+5)*random(0.0)+1.0) ]
    end  {  Of Initialization Routine     } ;


procedure checkconsistancy (hypothesis,previousrow: row;
                                      var e: eval);
    label 1090;
    var j1,j2: 1..4;
    begin
        { Count blacks first }
        e.black := 0;
        for j1 := 1 to 4 do
            if hypothesis[j1] = previousrow[j1] then
                e.black := e.black + 1;
        { Now count whites }
        e.white := 0;
        for j1 := 1 to 4 do
            begin
                for j2 := 1 to 4 do
                    if (j1≠j2) and
                       (hypothesis[j1] ≠ previousrow[j1]) and
                       (hypothesis[j2] ≠ previousrow[j2]) and
                       (hypothesis[j1] = previousrow[j2]) then
                       begin
                           e.white := e.white + 1;
                           { Dummy wrong value }
                           previousrow[j2] := colorless;
                           goto 1090     { Exit J2 loop }
                       end;
            1090:
            end
    end  { Of Check Consistancy Procedure }
```

of two components, *black* and *white*, each an integer between 0 and 4.

The variable *version* represents the version number, either 1 or 2. The 10 possible rows of code pegs in the game are recorded in the Pascal structure declared as:

var *rows:* array *[1..10]* of *row;*

Note that the careful selection of data representation makes the program much more clear and concise. The ability to deal with structures as a whole instead of just their elements tends to tighten up the logic of the program. For example, the BASIC lines:

```
820 REM ASSIGN NEXT ROW
830 FOR J=0 TO 3
840 LET R$(I+1,J)=D$(J)
845 NEXT J
```

are functionally equivalent to the Pascal assignment:

rows[i+1] := hyp {assign next row}

Also, the BASIC lines:

```
610 REM MAKE SURE THAT
    HYPOTHESIS ROW DOESN'T
    DUPLICATE ROW 1
620 LET Z=0
630 FOR J=0 TO 3
640 IF R$(0,J)<>D$(J) THEN 660
650 LET Z=Z+1
660 NEXT J
670 IF Z=4 THEN 700
690 GO TO 820
```

are functionally equivalent to the Pascal statement:

if *hyp* <> *rows[1]* **then goto** 820

Mr Milligan's BASIC version is well written and well structured. It contains three key routines: initialization (lines 50 to 210); generate hypothesis (lines 380 to 845); and evaluate response (lines 910 to 1100). However, due to the inexpressiveness of BASIC, it takes careful study, even of this well-written BASIC program, to recognize its structure. On the other hand, looking at the Pascal version of the same algorithm, the expressiveness of the language shows the structure at a glance. Similarly, the use of meaningful variable names and Pascal record structures makes the data representation readable. Table 1 describes which variables in the Pascal version are used in the same

context as variables in the BASIC version.

As careful as you are when coding BASIC, bugs are bound to creep in. For example, in the BASIC version (listing 4), lines 610 thru 690 are unnecessary. Additionally, there is no path through lines 770 to 810. Coding errors rarely creep into Pascal programs because the compiler enforces variable declarations and type agreement. For example, *evaluations[5] := rows[5]* is illegal because they are not type-compatible. Also *c := brown-red* is illegal because arithmetic is undefined for our user defined *colors* type. And, *version := 3* is illegal because the value 3 is outside the legal range for *version*.

Other Pascal Attributes

We have looked at some of the noteworthy features in Pascal. There are also the powerful features of block structured scope of names, recursion and dynamic allocation of storage. Pascal is known as a very "safe" language because it optionally has extensive compile and run time type checking including type compatability, subrange bounds and

BIBLIOGRAPHY

Introductory books on Pascal:

Bowles, K L, *Microcomputer Problem Solving With Pascal,* Springer-Verlag, New York, 1977.

Grogono, P, *Programming in Pascal,* Addison-Wesley, Reading MA, 1978.

Schneider, G et al, *An Introduction to Programming and Problem Solving With Pascal,* Wiley, New York, 1973.

Wirth, N, *Systematic Programming — An Introduction,* Prentice-Hall, Englewood Cliffs NJ, 1973.

Other books:

Dahl, O J, Dijkstra, E W and Hoare, C A R, *Structured Programming,* Academic Press, New York, 1972.

Jensen, K and Wirth, N, *Pascal User Manual and Report* (second edition), Springer-Verlag, New York, 1976.

Wirth, N, *Algorithms + Data Structures = Programs,* Prentice-Hall, Englewood Cliffs NJ, 1976.

The *Pascal Newsletter* is published quarterly by the Pascal Users Group for $4 per year. Contact Andy Mickel, University of Minnesota Computer Center, 227 Exp Engr, University of Minnesota, Minneapolis MN 55455.

```
function formhypothesis: Boolean;
   label 820;
   var i1,i2,i3,i4: colors;
      r: 0..9;
      hyp: row;
      eval1: eval;
      viable: Boolean;
   begin    { forming Hypothesis }
      viable := true;
      for i1 := last[1] to maxcolor do
      for i2 := last[2] to maxcolor do
      for i3 := last[3] to maxcolor do
      for i4 := last[4] to maxcolor do
         begin
            last := redrow;
            hyp[1] := i1; hyp[2] := i2; hyp[3] := i3; hyp[4] := i4;
            { Check all rows so far for consistancy }
            r := 0;
            repeat
               r := r + 1;
               checkconsistancy(hyp,rows[r],eval1);
            until (eval1 ≠ evaluations[r]) or (r = i);
            if eval1 = evaluations[r] then
               { Make sure that hypothesis doesn't duplicate row 1;
                 if it hasn't then we have a viable hypothesis      }
            if hyp ≠ rows[1] then goto 820;
               { Otherwise, keep searching....NEXT i4,i3,i2,i1 }
         end;
      viable = false;        { No viable hypothesis left }
      820:  if viable then
         begin    { Do not recheck eliminated possibilities }
            last := hyp;
            rows[i+1] := hyp        { Assign next row }
         end
      else begin
            writeln(' I HAVE REACHED AN IMPASSE. ');
            writeln(' COULD YOU HAVE MADE AN ERROR?')
         end;
      formhypothesis := viable       { Return with function value }
   end    { Of Form Hypothesis Procedure     } ;


begin      { Mastermind Codebreaker }
   repeat
      initialize;
      { Start main play of game here }
      for i := 1 to 9 do
         begin
            writeln;    write ('MY MOVE FOR ROW',i: 2,' is');
            for j := 1 to 4 do
               write (name[rows[i,j]]: 8); writeln;
            writeln ('HOW MANY BLACK PEGS?');
            read (evaluations[i].black);
            if evaluations[i].black = 4 then
               begin
                  writeln ('THANKS FOR THE GAME'); goto 870
               end;
            if evaluations[i].black = 3 then
                  evaluations[i].white := 0
            else begin
                  writeln (' HOW MANY WHITE PEGS?');
                  read (evaluations[i].white)
               end;
            if not formhypothesis then goto 870
         end;
      writeln (' I AM STUMPED —— YOU WIN!');
870:  repeat
         writeln (' ANOTHER GAME?'); read (ch)
      until (ch = 'Y') or (ch = 'N')
   until ch = 'N'
end    {   Of Main Program }
```

array index bounds. Pascal has many other data representations not illustrated here, such as sequential files, arrays, pointers and sets. I can't begin to explain all of these features here, but you don't have to understand all of them before you write your first Pascal program.

The main selling feature of Pascal is that properly developed programs are extremely easy to debug. Once you get a clean compile, the program usually runs! Why? Because the algorithms are expressed *clearly and naturally*. The range of all control variables are well specified and can be enforced at run time. The data types all agree and are appropriate to the problem. The program is readable — data types mean what they say — and it is therefore maintainable. Pascal encourages the methodical and systematic development of algorithms, an important structured programming method.

I hope this survey of Pascal has whet your appetite for the language. If so, read more about Pascal in this issue, then pick up any of the books in the bibliography and dive in!

Pascal is a rich and fertile language that emphasizes the expression of algorithms and data representation naturally and clearly. When will your microcomputer speak Pascal?■

BASIC Version	Pascal Version
Lines 220 to 270 and 850 to 900	**program** *mm2*
I DIM R$(9,3) R$ DIM S(9,1)	*i: 1 . . 9; j: 1 . . 4* *rows:* **array** *[1 . . 10]* **of** *row* *ch: char* *evaluations:* **array** *[1 . . 10]* **of** *eval*
Lines 50 to 210	**procedure** *initialization*
J DIM A$(6) V	*i: 1 . . 4; c: colors* *name:* **array** *[colors]* **of** *string* *color:* **array** *[0 . . 7]* **of** *colors* *version: 1 . . 2*
Lines 380 to 845	**procedure** *formhypothesis*
I0,I1,I2,I3 L0,L1,L2,L3 V DIM A$(6) DIM D$(3) R J N,M	*i1,i2,i3,i4: colors* *redrow, last: row* *maxcolor: orange. . space* *hyp: row* *r: 0 . . 9* *eval1: eval*
Lines 910 to 1100	**procedure** *Checkconsistency*
J1,J2 DIM C$(3) DIM B$(3) N,M	*j1,j2: 1 . . 4* *hypothesis: row* *previousrow: row* *e: eval*

Table 1: A comparison of the variables used in the two versions of the Mastermind game (see listings 4 and 5).

What Is Mastermind?

One of the most interesting conventional (ie: noncomputer) games on the market is "Mastermind," distributed by Invicta Plastics, Suite 940, 200 5th Av, New York NY 10010, and available in many local stores. Mastermind involves deductive logic, hypothesis testing and probabilistic inference. In Mastermind, the players take turns as "codemaker" and "codebreaker." The codemaker sets up a concealed row of four colored pegs from a set of Red, BLue, BRown, Green, Yellow and Orange pegs. It is acceptable to use the same color or colors more than once. In version 2, a more advanced game, empty Spaces are also permitted.

To challenge the computer program you are the codemaker. Write down a code. A row of four colors invokes the codebreaker computer program. It will take up to ten tries (rows) to discover the secret arrangement of colors in the concealed row. After printing each guess, the program will prompt you for the number of black and white key pegs.

The number of black pegs corresponds to the number of correct colors in correct positions. An important rule is that no position in the try is counted more than once.

When evaluating the program's try it is necessary to count black and white pegs carefully. If you make a mistake counting the number of exact or inexact correspondences, the program may exhaust all possible arrangements without finding a possible valid try. In this event the message:

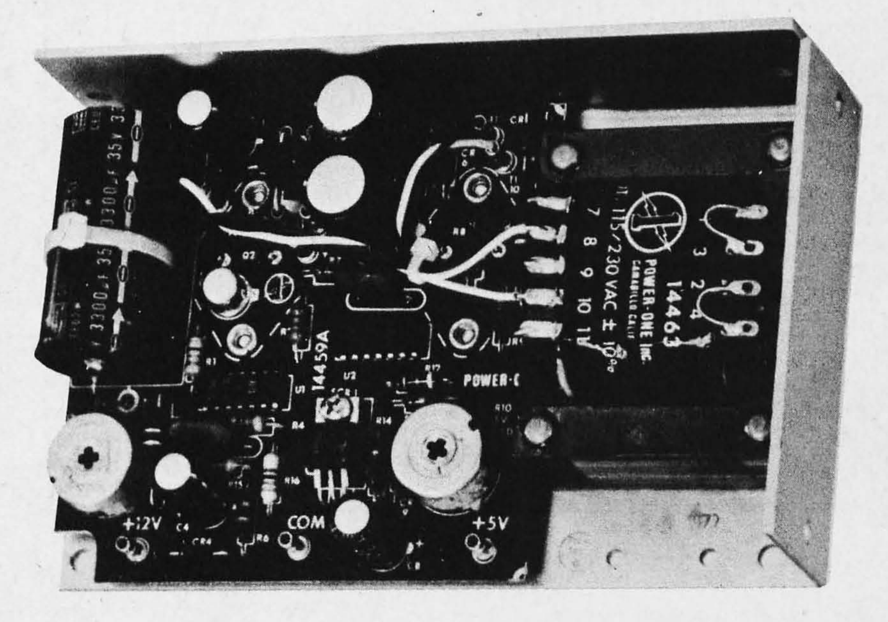
```
I HAVE REACHED AN IMPASSE.
COULD YOU HAVE MADE AN ERROR?
```

is printed.

(Adapted from W Lloyd Milligan's article, "Mastermind," October 1977 BYTE, page 168.)

What's New?

New Power Supply Dedicated to Minifloppy Format



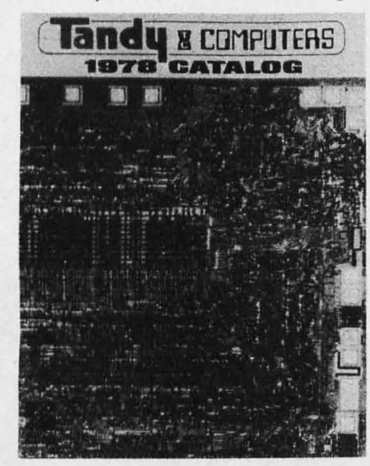
A new multiple output power supply, the CP-249, is, according to the manufacturer, the first power supply to be dedicated to the 5.25 inch floppy disk format.

The CP-249 provides dual outputs of 5 V at 0.7 A and 12 V at 1.1A steady state, with 1.7 A peak. The 5 V output includes overvoltage protection. Standard features include 115/230 VAC ±10% AC input capabilities ±0.05% line and load regulation, and full protection against short circuit and overload. Maximum ripple is 3.0 mV peak to peak, while transient response is 30 μs for a 50% load change. Operating temperature is rated at 0°C to +50°C at full ratings, derated to 40% at 70°C.

Single quantity pricing is $39.95. The unit is available from Power-One Inc, Power One Dr, Camarillo CA 93010.■

Circle 611 on inquiry card.

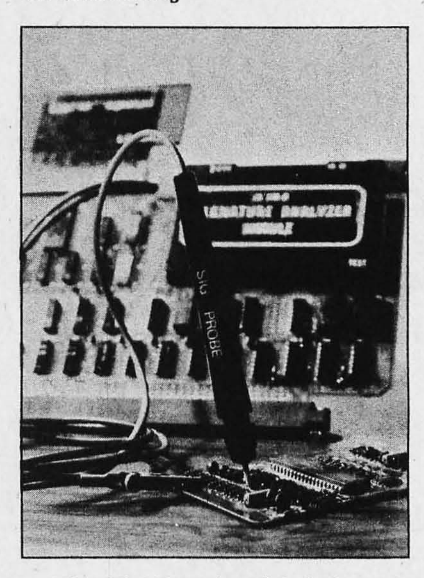
New Tandy Computers 1978 Catalog



A microcomputer mail order catalog has just been issued by Tandy Computers, Dept R7, POB 2932, Fort Worth TX 76101. The 52 page, 4 color catalog details a full line of popular brand microcomputers and accessories, software packages, parts and literature currently in stock. Kits and fully assembled microcomputer systems listed in the catalog range in price from several hundred dollars to more than $20,000. Among the nationally known brands carried by Tandy Computers are Radio Shack's TRS-80, the IMSAI 8080, Vector 1 and 1+, Xitan, Equinox 100, and PolyMorphic System 8813. Copies of the new catalog are available by telephoning toll free (800) 433-1679 or by writing the company.■

Circle 613 on inquiry card.

LS-100 Series Designed for Electronic Troubleshooting



The LS-100 series digital signature analysis product family provides the foundation for LSI and microprocessor electronic troubleshooting and repair.

Verification of correct digital patterns provides go and no-go testing as well as diagnostics. The series provides for rapid identification of bad components, printed circuit boards and entire systems. Options include remote LED signature display, 32 line multiplexer, logic probes, enhanced software package and stand alone test ability.

The LS-100 expandable series is available in S-100, LSI-11, or EXORciser bus compatible plug ins, ready to go. Prices start at $295. Contact Phoenix Digital Corp, POB 11628, Phoenix AZ 85017.■

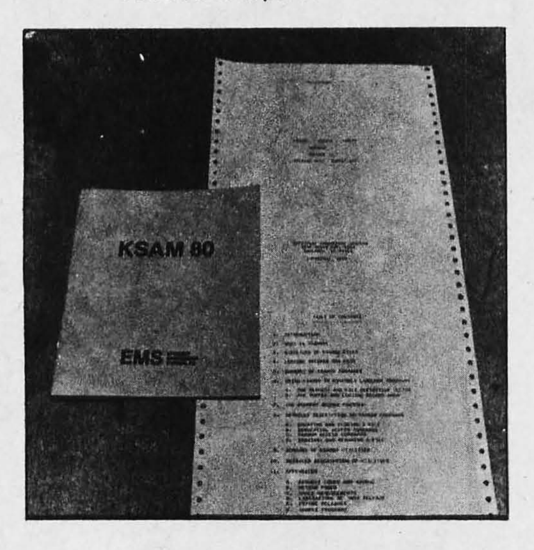
Circle 612 on inquiry card.

AVA Introduces New RF Video Adapter Line

AVA Electronics announces the introduction of their new RF (radio frequency) Video Adapter line. It features over 40 adapters and accessories suitable for home and professional video applications. The adapters cover phono, BNC, UHF and F connectors. The free #478 catalog sheet describes these products in detail and is available from AVA Electronics Corp, 242 Pembroke Av, Lansdown PA 19050.■

Circle 614 on inquiry card.

File Management System for Floppy Disk Microcomputers



KSAM80 is a file management system designed specifically for floppy disk microcomputer systems. It was originally developed under Zilog's Z-80 OS 2.0 but can be implemented in many existing microcomputer operation systems.

KSAM80 was developed for applications such as inventory control, reservation systems, library systems, accounts receivable and bill of materials processing. Random storage and retrieval of records is based on the contents of a user defined data field within the record which is called the key. The key must be unique for each record and can be any string up to 255 characters long.

KSAM80 also supports sequential access of records starting at any point within a file, random access by partial key and random access by relative record number. Sequential and random access commands can be intermixed freely.

Space is automatically allocated to the file when records are added and reclaimed when records are deleted so that files are self-reorganizing and any number of files can be processed simultaneously provided that sufficient buffer storage is available.

A number of utility programs are available as part of the KSAM80 package. For additional information write to EMS, 3645 Grand Av, Suite 304, Oakland CA 94610.■

Circle 532 on inquiry card.

A New Macroassembly Language for 8080 and 8085 Microprocessors

SMAL/80 is a compiled, structured, macroassembly language for 8080 and 8085 microprocessors that requires only 7 K bytes of memory. SMAL/80 statements are written in a symbolic notation resembling PASCAL and PL/M that simplifies the writing of assembly language programs. It incorporates the basic structured programming constructs, the do-end, if-then-else, and loop-repeat, which may be combined with and/or nested within each other without limit to form highly complex statements.

The package includes a 2 K byte macropreprocessor written in SMAL/80. The macropreprocessor permits conditional expansion of statements, and unlimited nesting of macros.

Also included in the SMAL/80 package is a translator program that allows one to convert any 8080 or 8085 program written in standard Intel mnemonics into SMAL/80 without the constructs.

SMAL/80 is being offered initially in CP/M and Isis I disk formats; the price, including documentation, is $75. This software is available from Chromod Associates, POB 3169, Grand Central Station NY 10017.■

Circle 535 on inquiry card.

Business Software In North Star BASIC

Available now is a series of three business software packages in North Star BASIC. Features of Micro-Base Inventory are said to include multiple key ISAM (indexed sequential access method) disk structuring; inquiry keys by catalogue number, manufacturer, or item description; multiple selling price capability for each item; formatted screen handling; full retail or cost extension reporting of items on hand or on order or sold. The second module, Micro-Base Accounts Payable, is described as a fully featured data base management system, with real time inquiry for file information. Features of this system are multiple key ISAM disk structuring, inquiry keys by vendor number, vendor name, voucher number; formatted screen handling; paid voucher history file; bank account reconciliation assistance; cash requirements forecasting. Micro-Base Accounting system is said to include payroll, time and material billing, purchase ordering, accounts receivable, accounts payable, inventory handling, and general ledger and financial reporting. The price for these systems is $450 per module from Computertex, POB 66907, Houston TX 77006.■

Circle 537 on inquiry card.

Character Oriented Processing System for the Z-80 or 8080 Printer

The Electric Pencil II is a character oriented processing system. Text is entered as a continuous string of characters and is manipulated as such, allowing the user freedom and ease in the movement and handling of text.

Features of the Electric Pencil II include CP/M compatibility, disk operating system which supports two disk drives, file management, disk storage and retrieval, print formatting multicolumn printing, print value chaining, page at a time scrolling, bidirectional multispeed scrolling controls, subsystem with print value scoreboard, automatic word and record number tally end of page control.

Hardware must include a microcomputer using the 8080 or Z-80 microcomputer, printer, video display (VDM-1, VII or SOL) CP/M supported disk system or North Star minifloppy disk or cassette interface (Tarbell or SOL).

The Electric Pencil II is available on CP/M. The price for the standard printer version is $225 and the Diablo printer version is $275. For further information, contact Michael Shrayer Software, 3901 Los Feliz Blvd, Los Angeles CA 90027.■

Circle 536 on inquiry card.

Super Startrek

Super Startrek requires a terminal equivalent to the Soroc IQ120 or the ADM-1 (with screen protect, screen clear and cursor control) with 48 K bytes of memory.

A sector map, status display and galactic map are placed on the screen in protected areas. It includes star bases, the USS Enterprise, the Faire Queen, Klingons, command Klingons and cloaked Romulans.

Examples of operational commands are WAR (warp: sets warp factor for moves), MOV (move: direction and distance), IMP (impulse engines: allow one sector moves), and ABA (abandon ship to Faire Queen).

Super Startrek is written in North Star BASIC with complete, comprehensive playing instructions. The price is $51 and the software is available from Aaron Associates, POB 1720A, Garden Grove CA 92640.■

Circle 533 on inquiry card.

APL Colloquium Series

Think Inc, a company which develops and markets APL application packages, is offering a monthly colloquium series on APL topics. Topics such as "A Proposed Standard for the Interchange of APL Workspaces" and "The Use of APL for Systems Work" have been discussed in previous series. Think Inc plans to continue the colloquium series until the summer. For further information contact Brooke Tompkins, Think Inc, 310 E 46th St, New York NY 10017.■

Circle 534 on inquiry card.

FOR ALL CUSTOMERS EXCEPT CALIF. CALL TOLL FREE 800-421-5809

21L02 (350ns) Static Rams 120 @ $1.00 ea.	1702A E-PROM $4.75 ea.	6502 Microprocessor 5 @ $11.00 ea.	2708 (450ns) E-PROM 8 @ $9.00 ea.	21L02 (250ns) Static Rams 100 @ $1.25 ea.	Z-80A Microprocessor 5 @ $25.00 ea.	8212 8 Bit I/O Port 25 @ $3.00 ea.	4116 (200ns) 16K Dyn. Ram 8 @ $24.00 ea.
Z-80 Microprocessor 5 @ $20.00 ea.	8224-4 Clk.Gen.&Dvr. 25 @ $8.75 ea.	410D (200ns) Static Ram 100 @ $9.50 ea.	TMS 4044 (250ns) 16 @ $8.95 ea.	4200A (200ns) Static Rams 25 @ $10.00 ea.	74LS367 Hex Buffer 100 @ .70¢ ea.	74LS368 Hex Inverter 100 @ .70¢ ea.	2513 (5v) Character Gen. 5 @ $9.00 ea.

4901 W. ROSECRANS, HAWTHORNE, CA. 90250

MICROCOMPUTER COMPONENTS

MICROPROCESSOR'S

F8	16.95
Z80	22.00
Z80A	28.00
CDP1802CD	19.95
2650	24.95
AM2901	22.95
6502	11.95
6800	18.95
6802	25.00
8008-1	9.95
8035	22.00
8080A	11.95
8085	27.00
TMS9900TL	75.00

8080A SUPPORT DEVICES

8212	3.50
8214	9.00
8216	3.75
8224	3.50
8224-4	9.95
8226	3.95
8228	7.95
8238	7.50
8251	9.95
8253	21.95
8255	21.95
8257	21.95
8259	21.95
8275	75.00
8279	20.00

MISC. OTHER COMPONENTS

N8T20	3.39
N8T26	2.10
N8T95	1.35
N8T96	1.35
N8T97	1.35
N8T98	1.35
81LS95	2.00
81LS97	2.00
1488	1.75
1489	1.75
D3205	4.00
D3207A	4.55
D3208A	14.20
D3211	10.00
B3222	9.75
B3242	10.15
D3245	5.60
C3404	6.75
P3408A	12.00
P4201A	5.20
MM5320	7.50
MM5369	1.90
TMS5501	24.95
DM8130	2.90
DM8131	2.75
DM8833	2.50
DM8835	2.50
DM8837	1.75
MK50240	20.00
MK50250	15.00

FLOPPY DISC CONTROLLER

1771B	55.95
1771B-01	57.95

KEYBOARD CHIPS

AY5-2376	13.75
AY5-3600	13.75

PROM'S

1702A	4.00
2704	15.00
2708	12.00
2716	30.00
2716 Intl	38.00
2758	26.60
D3601	4.50
D3604	13.00
5203AQ	5.00
5204AQ	7.50
6834	17.50
6834-1	14.95
82S23B	4.00
82S129B	4.25
8223B	3.50

6800 SUPPORT

6810P	4.95
68B10P	6.00
6820P	7.50
6821P	7.50
6828P	11.25
6834P	16.95
6850P	9.75
6852P	11.75
6860P	10.00
6862P	14.50
6871P	28.00
6875P	8.75
6880P	2.50

Z80 SUPPORT DEVICES

3881	12.95
3882	12.95

CHARACTER GENERATORS

2513	6.75
2513 5v upper	9.75
2513 5v lower	10.95
2516	10.95
MCM6571	10.95
MCM6571A	10.95
MCM6574	13.25
MCM6575	13.25

WAVEFORM GENERATOR

8038	3.50
MC4024	2.25
566	1.50

DYNAMIC RAMS

416D/4116	32.00
1103	1.00
2104	4.00
2107B	4.25
2107B-4	3.95
TMS4050	4.00
TMS4060	4.50
TMS4070-2	32.00
4096	4.00
4116/416D	32.00
MM5270	4.50
MCM6605	5.00

USRT

S2350	10.75

UART'S

AY5-1013A	5.25
AY5-1014A	8.25
TR1602B	5.25
TMS6011	5.95
IM6402	10.80
IM6403	10.80

STATIC RAMS

	1-16	17-63	64 +
21L02	1.50	1.24	1.18
21L02 (350)	1.60	1.35	1.25
21L02 (250)	1.75	1.60	1.50
410D	10.75	10.00	9.25
1101A	1.00	.90	.80
2101-1	2.95	2.75	2.60
2102	1.25	1.15	1.00
2111-1	3.95	3.50	3.25
2112-1	2.95	2.80	2.69
2114-3	11.00	10.00	9.25
2125L	11.10	9.00	8.30
2147	37.50		
31L01	2.50	2.35	2.00
3106	3.95	3.70	3.25
3107	3.95	3.70	3.25
TMS-4044	9.95	9.00	8.95
4200A	12.95		
TMS-4045	11.00	10.00	9.25
5101	8.30	7.40	7.25
74C89	3.25	3.05	2.85
7489	2.25	2.10	1.90
74S201	4.50	4.00	3.75
P8101	4.20	3.40	2.80
P8155	17.00	14.00	
P8156	21.00	18.00	
8599	1.88	1.75	1.60
9102BPC	1.65	1.45	1.30

TV-1

Convert T.V. set to Video Monitor.

KIT $8.95

JADE 8080A KIT

$100.00 KIT

BARE BOARD $30.00

MEMORY PLUS

for KIM-1
8K RAM (21L02)
8K EPROM
ASSEMBLED & TESTED
$245.00

KIM-1

ASSEMBLED & TESTED $245.00

MODEM

80-103A Serial I/O and FSK modem for professional and hobby communications.

*Completely compatible with your IMSAI, ALTAIR, SOL, or other S-100 microcomputers.
*Designed for use on the dial phone or TWX networks, or 2-wire dedicated lines, meets all FCC regulations when used with a CBT coupler.
*All digital modulation and demodulation with on board crystal clock and precision filter mean that NO ADJUSTMENTS ARE REQUIRED.
*Bell 103 standard frequencies
*Automated dial (pulsed) and answer
*Originate and answer mode
*110 or 300 BPS speed select
*Complete self test capability
*Character length, stop bit, and parity
*90 day warranty and full documentation.

PRICES:

Bare Board & Manual **$49.95**
Assembled (48hr burn) **$279.95**
JG-DCA Kit **$159.95**

full ASCII PROFESSIONAL KEYBOARDS

* Full 128 Character ASCII
* Tri-Mode MOS Encoding
* MOS DTL TTL Compatable Output
* Two-key Rollover
* Level and Pulse Strobe
* Shift and Alpha Lock
* Selectable parity
* Positive or Negative Logic.

MODEL 756 (56 keys)

PRICING INFORMATION

Model 756 (assembled)	$59.95
Model 756K (kit)	$49.95
Model 702 enclosure	$29.95
Model 710 Numeric Pad	$9.95
Model 756MF Mtg.Frame	$8.95

MOTHER BOARD's - S-100 STYLE

13 slot with front panel slot
BARE BOARD $35.00
KIT $80.00
22 Slot $149.95
Assembled & Tested

CONNECTORS

DB-25P **$2.25** DB-25S **$3.25**
COVER **$1.50**

44 Pin - PC & EYE	$1.95
44 Pin - WW	$2.50
86 Pin - (6800) PC	$5.00
86 Pin - (COSMAC ELF) PC	$5.00
100 Pin - (Altair) PC	$4.50
100 Pin - (Imsai) WW	$4.25
100 Pin - (IMSAI) PC	**$3.25**

JADE Z80 KIT

—with PROVISIONS for ONBOARD 2708 and POWER ON JUMP

$135.00 EA. (2MHZ)
$149.95 EA. (4MHZ)
BARE BOARD **$35.00**

JADE VIDEO INTERFACE KIT

FEATURES **$99.95**

S-100 Bus compatible
32 or 64 Characters per line - 16 lines
Graphics (128 x 48 matrix)
Parallel /4 compositive video
On board low-power memory
Powerful software included for cursor, home, EOL, Scroll Graphics/Character Upper case, lower case and Greek. Black-on-white & White-on-black.

JADE PARALLEL/SERIAL INTERFACE KIT

$124.95 KIT

* S-100
* 2 Serial interfaces with RS232 interfaces or 1 Kansas City cassette interface.
* Serial interfaces are crystal controlled
* Selectable baud rates.
* Cassette works up to 1200 baud.
* 1 parallel port.

STATIC RAM BOARDS

ASSEMBLED & TESTED

8K

Ram 8 (250ns)	$169.95
Ram 8B (450ns)	$139.95
250ns KIT Mem-1	$169.95
450ns KIT Mem-1	$125.00
Bare Board	$25.00
6800 Adapter for S-100 system for Mem-I only KIT	$12.95

16K Uses 2114L

Ram 16 (250ns)	$375.00
Ram 16B (450ns)	$325.00

32K

JG-32K (250ns)	$875.00
JG-32KB (450ns)	$775.00
450ns KIT	$675.00

DYNAMIC RAM BOARD

by S. D. Computer Products

On board refresh is provided with no wait states or cycle stealing required. +8VDC 400MA DC, +18VDC 400MA and -18VDC 30MA DC.

EXPANDABLE 32K (uses 4115) 200ns
8K Kit $151.00 24K Kit $367.00
16K Kit $259.00 32K Kit $425.00

EXPANDABLE 64K (uses 4116) 200ns
16K Kit $281.00 48K Kit $757.00
32K Kit $519.00 64K Kit $995.00

16K STATIC BOARD

with memory management can be used with Alpha Micro or Cromenco Systems.

RAM 65(250ns)	$350.00
RAM 65B (450ns)	$390.00

COMPUTER MAINFRAME

Includes: **$295.00**
Power Supply +8v at 18amps
+16v at 2 amps
Mother Board - 12 slots with connectors Assembled & Tested
Has Whisper Quite Fan & AC line fitler
Cabinet size 7"H x 19"W x 22" D

THE PROM SETTER

WRITE & READ EPROM

1702A - 2708 - 2716 - 5204 - 6834

*Plugs directly into your ALTAIR, IMSAI computer.
*Includes main module board and external EPROM socket unit.
*The EPROM socket unit is connected to the computer through a 25 pin connector.
*Programming is accomplished by the computer.
*Just read in the program to be written on the EPROM into your processor and let the computer do the rest.
*Use socket unit to read EPROM's contents into your computer.

KIT $210.00

ASSEMBLED $375.00

S-100 Power Supply with Cabinet

Power supply +8v at 18 amps
±16v at 2 amps
Has connector for Power output
MODEL UPS=600 size 5"Hx8"Wx6½"D
PRICE $149.95

E-PROM BOARDS

MR-8 (8K uses 2708) KIT with 1K RAM	**$99.50**
MR-16T (16K uses 2716) KIT with 1K RAM	**$99.50**
MM-16 (16K uses 2708)	**$99.00**
RAM/N/ROM (16K uses any E-PROM) KIT	**$117.00**
JG-8/16 (uses 2708 or 2716) KIT	**$59.95**
BARE BOARD	**$30.00**

EXPANDABLE E-PROM — S.D.Sales

16K or 32K EPROM **$49.95** without EPROM

Allows you to use either 2708's for 16K of Eprom or 2716's for 32K of EPROM.

FLOPPY DISC INTERFACE

JADE Floppy Disc (Tarbell Board) KIT **$175.00 ea.**

S.D. Sales Versa Floppy Kit **$149.00 ea.**

MODEL 801R Shugart with Disc Cabinet

Includes Cabinet, Disc. Drive, Power Supply, Cable, Fan & Data Cable. Has AC line filter.
Cabinet size 10"H x 10"W x 16"D
MODEL DM 2700-S **$835.00 ea.**

JADE

Computer Products

RETAIL STORE HOURS: Monday - Friday 9—7
Saturday 9—5

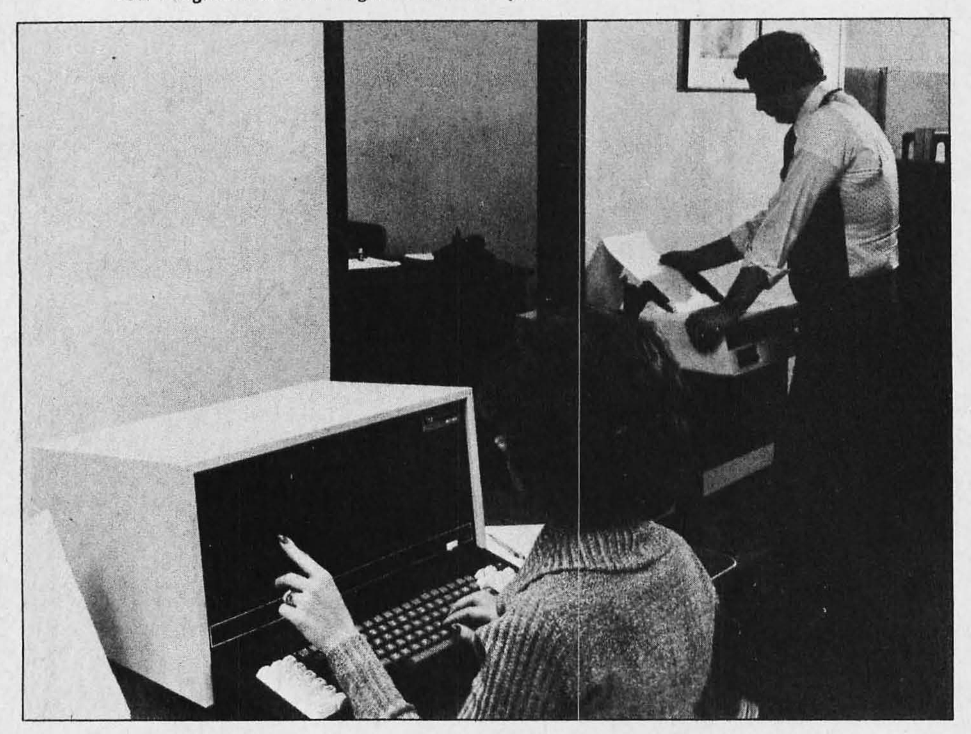
Discounts available at OEM quantities ADD $1.50 under 10 lbs. for shipping. California residents add 6% sales tax.

NEW CATALOG NOW AVAILABLE

VISA
AMERICAN EXPRESS
Cards Welcome
master charge

Circle 195 on inquiry card.

User Programmable Intelligent Terminal System

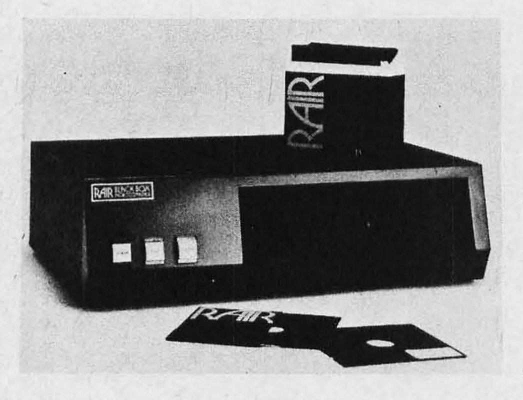


Fully user programmable, the Mini Disk Terminal (MDT) 400 is an intelligent terminal system which can be used for a variety of distributed data processing applications including data entry, test processing and data communications. The MDT-400 features: minidiskette; 8 bit microprocessor and communications interface; high resolution video display with 16 line by 80 word format, scrollable through 2560 character buffer; 122 key solid state keyboard with 18 user defined keys and loadable formats; integral minidisk with 87 K byte capacity in IBM soft sector format; low and medium speed character printers for hardcopy output; asynchronous and binary synchronous communications interfaces for line speeds up to 9600 bps; program development systems supporting assembler and high level language and text processing applications package.

Representative prices include: Model 401 with 8 K bytes programmable memory, integral minifloppy, video and keyboard is priced at $4275; Model 402 with 15 K bytes programmable memory, integral minifloppy, video and keyboard is priced at $4500; Model MDT-441 low speed, 60 cps, matrix printer is priced at $2495. For further information, write to Compugraphic, 80 Industrial Way, Wilmington MA 01887.■

Circle 543 on inquiry card.

Floppy Disk Microcomputer Supported by Custom Software



An integrated microcomputer from Britain incorporates floppy disk drives and software. The Rair Black Box dual disk drives accept 5¼ inch (13.3 cm), single sided, reversible, soft sectored floppy disks, each with a capacity of 81.92 K bytes. Each disk has 40 tracks with 16 sectors per track. Transfer rate is 125,000 bps; latency is 100 ms; and access time is 40 ms track to track.

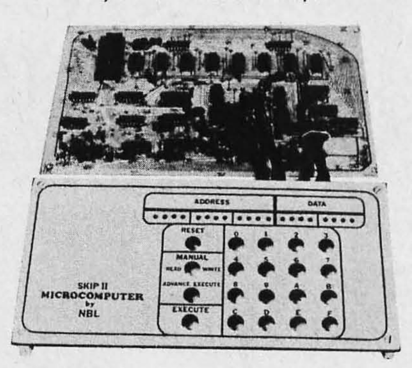
The microprocessor handles up to 80 instructions; cycle time is 1.28 μs. The unit is housed in a self-contained desktop cabinet.

The programmable memory contains 65.536 K bytes. Software includes a disk operating system, an extended BASIC interpreter, a relocatable FORTRAN compiler, and a COBOL compiler. Other features include dual serial IO ports and an 8 slot mother board for system expansion.

For further information contact Rair, 30-32 Neal St, London WC2H 9PS ENGLAND.■

Circle 544 on inquiry card.

Low Cost, Do It Yourself Computer



A new do it yourself computer called SKIP II has 1 K bytes of programmable memory, LED display of both address and data, hexadecimal keyboard, screen printed front panel, printed circuit board and National Semiconductor's SC/MP computer chip. It can be built for less than $100. This includes detailed instructions, system checkout guide and troubleshooting manual. A programming guide illustrates all of the 46 instructions with simple programs which explain what the computer is doing at each stop. It is said to be ideal for the beginner. Contact NBL, POB 1564, Richardson TX 75080.■

Circle 545 on inquiry card.

Two New F8 Products for the Personal Computer User

The introduction of two new F8 products for the personal computer user and design engineer has been announced by Comptronics, 19824 Ventura Blvd, Woodland Hills CA 91364.

The F-8S100 is a processor board compatible with the S-100 bus. The unit provides sockets for 2 K bytes of erasable read only memory monitor, two PIO sockets and connections for six IO ports. The board has 64 bytes of scratch pad programmable memory and a fully buffered data bus. The Model F-8S100 sells for $239 as a kit or $275 assembled.

The second product is an F8 microcomputer, Model KD80, with keyboard and 6 digit display. The unit provides audio interface and speaker compatible with the onboard KD-BUG (3856) music routine, 2 K bytes of random access memory expandable through an S-100 connector, and 1 K bytes of erasable read only memory with four additional 2708 sockets. Model KD80 sells for $375 as a kit and $425 assembled.■

Circle 546 on inquiry card.

California Industrial

Post Office Box 3097 B • Torrance, California 90503

Hazeltine 1400



cost effective

CRT TERMINAL

$659 plus shipping

The Hazeltine 1400 Vidio Display Terminal is designed to optimize interactive real-time operations. The interface is capable of either local or remote connection through an EIA RS232-C interface at baud rates that are switch selectable up to 9600 baud.

- All 128 ASCII Codes
- 64 Displayable Characters
- 24 Lines; 12 inch Screen
- 80 Characters per Line
- Self Diagnostic Test

Quantity pricing upon request.

S-100 Mother Board

Quiet Buss™

$29.95

18 slot IMSAI

The Quiet Buss from California Industrial is quality engineered. No short cuts have been taken to produce this mother board. Active termination circuitry prevents noise and crosstalk. Manufactured from extra heavy FR-4 epoxy glass.

TELETYPE® MODEL 43



New from Teletype, the Model 43 is capable of printing 132 ASCII characters per line. Send and receive data at 10 or 30 char. per second. Keyboard generates all 128 ASCII code combinations. RS-232 interface, same as the popular Model 33. Data sheet sent upon request. Manufacturer suggested price $1377.00.

IMMEDIATE DELIVERY $1219

TTL model with NOVATION brand Acoustic Modem. $1419

TELETYPE

HEXADECIMAL KEYBOARD

$34.95

Maxi-Switch hexadecimal keyboards are designed for microcomputer systems that require 4-bit output in standard hex code.

Each assembly consists of 16 hermetically sealed reed switches and TTL "one shot" debounce circuitry.

Reliable low friction acetal resin plungers are credited for the smooth operation and long life of this premium keyboard.

Requires single +5 volt supply.

NEW MAXI-SWITCH ASCII KEYBOARD



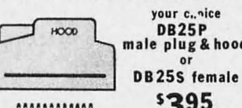
Fully Assembled **$64.97**

Definitely the best small system keyboard that we have seen. Maxi-Switch has incorporated all the important keyboard features at a reasonable price. Full 128 ASCII functions, "N" key rollover, automatic repeats, user designated special function keys, escape, control & lots of others. Data sheet upon request.

CONNECTORS

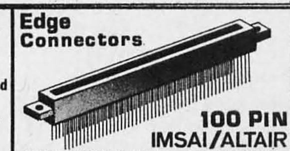
your choice **DB25P** male plug & hood or **DB25S** female

$3.95



Qty.	fe.	male	hd.
10	3.45	2.45	1.15
25	3.15	2.25	1.05
100	2.85	1.90	.95
500	2.25	1.60	.85
1K	1.97	1.37	.73

Edge Connectors



100 PIN IMSAI/ALTAIR

S-100 • GOLD PLATED • .125" CENTERS

Altair .140 row, soldertail..... $5.98 3/$16.50

Imsai .250 row, soldertail..... $4.98 3/$11.95

3 Level Wire Wrap .250 row... $4.98 3/$13.00

SPECIALS

W/W same as above without ears $3.50 3/$10

72 (dual 36) W/W .156" centers... $2.50 3/$6

SPERRY UNIVAC KEYBOARD



$24.88

The famous Sperry Univac 1710 Hollerith keyboard assembly is now available from California Industrial for only $24.88. The ideal computer input device for accountants and mathematicians. The numeric keys are placed on the lower three rows to resemble a ten key adding machine. This format allows one handed numeric data entry.

Original cost was $385. Used but guaranteed in excellent condition. Complete with documentation.

FROM ATARI COLOR TELEVISION R.F. MODULATOR

$13.95



Assembled

The Atari R.F. Modulator allows computer data to be displayed directly upon your existing television system. This unit converts the signal from the Apple II and other video sources into television frequencies. Operates from single 5 volt supply. Complete with metal case, mating R.F. connector and 15 feet of coax cable. Schematics and instructions included.

SPECIAL APPLE II

16K MEMORY

COLOR • GRAPHICS • SOUND

$1024 Mfg. Sug. Retail.... $1195

Scotch® BRAND

$4.98 10 for $45.

DISKETTES

- 8 inch Soft (IBM)
- 8 inch 32 sector
- Mini Soft sec.
- Mini 10 sector
- Mini 16 sector

Certified Digital CASSETTES

Won't drop a BIT!

$5.50



CALIFORNIA INDUSTRIAL is an Authorized Dealer of Scotch Brand Data Products

JOYSTICK $4.50



3 for $10.00

This joystick feature four 100K potentiometers, that vary resistance proportional to the angle of the stick. Perfect for television games, quad stereo and radio controlled aircraft.

2708



450ns **2708** **$9.95**

8K UV Erasable MEMORY

Potter & Brumfield REED RELAY

5 volt coil, pulls 3.5v.



$1.19 ea.

SPST	10	100	1K
	.97	.79	.65

S-100 PROTOTYPE BOARD

$19.98



GP100-Maximum design versatility along with standard address decoding and buffering for S100 systems. Room for 32 uncommitted 16 pin IC's, 5 bus buffer & decoding chips, 1 DIP address select switch, a 5 volt regulator and more.

WW100-Wire wrap breadboard, similar to the GP100. Allows wire wrap of all sizes of sockets in any sizes of sockets in any combination. An extra regulator position for multiple voltage applications.

21L02

LOW POWER 450 NS

$1.19 SUPER BUY! Lowest Price Anywhere

Our low power static RAMs are factory prime. Purchased, on contract, directly from one of California's leading semiconductor manufacturers.

Access time guaranteed faster than 450nS. Minimum purchase 32 pieces.

Sorry, credit cards can not be accepted on 21L02's.

7400	.13	7442	.59
7401	.19	7443	.79
7402	.19	7445	.89
7403	.19	7446	.99
7403	.19	7447	.99
7404	.19	7448	.99
7405	.19	7450	.25
7406	.19	7451	.25
7407	.25	7453	.25
7408	.25	7454	.25
7409	.25	7460	.25
7410	.19	7470	.25
7411	.25	7472	.39
7412	.35	7473	.39
7413	.49	7474	.35
7414	.79	7475	.49
7416	.39	7476	.39
7417	.39	7479	3.99
7420	.19	7480	.79
7422	.49	7482	.99
7423	.39	7483	.99
7425	.39	7485	.99
7426	.39	7486	.49
7427	.39	7488	3.40
7428	.49	7489	2.79
7429	.39	7490	.49
7430	.25	7491	.99
7433	.39	7492	.49
7437	.39	7493	.49
7438	.39	7494	.79
7439	.39	7495	.79
7440	.39		

7496	.79
7497	3.99
74100	1.19
74107	.39
74109	.49
74110	1.79
74116	1.99
74120	1.79
74121	.39
74122	.39
74123	.69
74125	.59
74126	.59
74128	.49
74132	.99
74136	.89
74141	.99
74145	.99
74147	2.49
74148	1.99
74150	1.19
74151	.99
74153	.89
74154	.99
74155	.99
74156	1.29
74157	.99
74159	2.99
74160	1.19
74161	.99
74162	1.49
74163	.99
74166	1.19
74167	4.99
74170	2.49
74173	1.49
74174	1.19
74175	.99
74176	.99
74177	.99
74192	.99
74193	.99

CMOS

4001	.25
4002	.25
4006	1.99
4007	.25
4008	1.49
4009	.69
4010	.69
4011	.25
4012	.25
4013	.49
4014	1.49
4015	1.39
4016	.69
4017	1.29
4018	1.69
4019	1.79
4020	1.39
4021	1.49
4022	1.25
4023	.25
4024	1.19
4025	.25
4027	.69
4028	1.25
4029	1.99
4030	.69
4032	.49
4033	1.99
4035	1.69
4040	1.99
4041	1.49
4042	1.49
4043	1.49
4044	1.49
4046	2.49
4047	2.49
4049	.79
4050	.79
4051	1.99

CPU's

8080A	14.95
6800	24.95
Z80	39.95

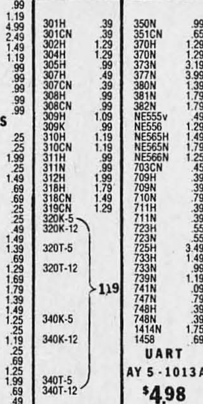
MEMORY

1702A	4.95
82s23	2.95
82s123	2.95
2102	1.79
2102-1	1.89
21L02	1.19
250nS	1.49

CLOCK's

5314	2.95
5316	4.95
5375	3.95

LM741 $ SOLD OUT



301H	.39
301CN	.39
302H	1.29
304H	1.29
305H	.99
307H	.49
307CN	.39
308H	.99
308CN	.99
309H	1.09
309K	.99
310H	1.19
310CN	1.19
311H	.99
311N	.99
312H	1.99
318H	1.79
318CN	1.49
319CN	1.29
320K-5	1.19
320K-12	1.19
320T-5	1.19
320T-12	1.19
340K-5	1.19
340K-12	1.19
340T-5	1.19
340T-12	1.19

350N	.99
351CN	.65
370H	1.29
370N	1.29
373N	3.19
377N	3.99
380N	1.39
381N	1.79
382N	1.79
NE555v	.49
NE556	1.29
NE565H	1.49
NE565N	1.79
NE566N	1.25
703CN	.45
709H	.39
709N	.39
710N	.79
711H	.39
711N	.39
723H	.55
723N	.55
725H	3.49
733H	1.49
733N	.99
739N	1.19
741N	.09
747N	.79
748H	.39
748N	.39
1414N	1.75
1458	.69

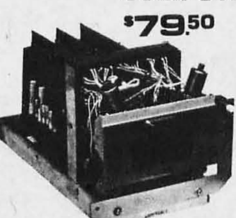
UART AY 5-1013A **$4.98**

SPECIAL

GENERAL INSTRUMENT ASCII Keyboard Encoder AY 5-3600 Prime but house marked only $4.95

Digital Cassette Drive

COMPUTER CONTROLED



$79.50

This precision I/O assembly features remote software controlled search capabilities. Two independent capstan drive motors allow the computer to control direction and speed of the transport.

The assembly consists of a Raymond cassette transport, chassis, motherboard and three edge cards: read/write, capstan drive & control card.

Current replacement valued at over $700.00. Schematics and complete documentation included. USED, but in excellent condition.

Thumbwheel switch



Ten position **BCD**

$1.39 ea.

10	50
$1.19	.89

MINIATURE SWITCHES

your choice

$.98

10	50	100	1k
$.88	.81	.73	.66

SPDT Miniature Toggles

- 7101 C&K ON-NONE-ON
- 7107 jbt ON-OFF(mnt.ON)
- 7108 CK ON-(moment. ON)
- Rocker JBT ______ DPDT
- Rotary 3P-4-Pos.
- Rotary 3P-6-Pos.
- Push B (N.O.) $.39ea. 4/$1

CAPACITORS

ELECTROLYTICS

	ea.	10	50
80,000/10v.	3.95	3.49	2.95
4500/50v.	$1.49	1.35	1.19
1000/15v axial	$.55	.49	.45
.1 disc	$.12	.09	.07
.01 disc	.06	.05	.04





DIP Switch

$1.49 ea.

10	25	100	1k
$1.29	1.15	.97	.85

specify 7 or 8 pos.



SPST CTS

50 Conductor RIBBON WIRE

$.69 Ft.

TWISTED PAIR

Power Adapter

- 6 vdc, 140mA $1.39
- 7 vdc, 1.4 A. 5.50
- 9 vdc, 15mA. 1.19
- 10 vAc, 300mA. 1.95

RELAYS

SPDT MINIATURE

$1.19 ea.

10	25	100
$1.15	1.04	.89

Coil 12 Volt dc.

7 Amp Contacts

P. C. Board Mount

Transistors

	ea.	10	50	100
2N2222A	.20	.18	.16	.15
2N3055	.69	.65	.59	.55
MJ3055	.79	.75	.69	.65
2N3772	1.59	1.49	1.39	1.29
2N3904	.15	.11	.09	.07
2N3906	.15	.11	.09	.07

Diodes

		10	25	100
1N4002	100v.	.08	06	.05
1N4005	600v.	.10	.08	.07
1N4148	signal	.07	.05	.04

jumbo red **LED's**

ea.	10	25	100
$.15	.13	.11	.09

BINDING POSTS 5-WAY

3 for $1.19

20	100
$.35	.29

9 foot

$1.49

Heavy duty grounded power cord and mating chassis connectors.

SOLDERLESS TERMINALS

INSULATED

20 for $.98

Specify: 22-18; 16-14

100	500	1k
4.50	20.	35.

PANASONIC

$.98

Nicad "AA" 450mA.

DISCOUNT Wire Wrap Center

IC SOCKETS

pin	wire wrap ea.	25	50	low profile ea.	25	50
8	—	—	—	17¢	16	15
14	37¢	36	35	18	17	16
16	38	37	36	19	18	17
24	99	93	85	36	35	34
40	169	155	139	63	60	58

KYNAR WIRE WRAP

50ft. $.98

500	1,000	11,000
$9.	$15.	$105.

Battery wire wrapping tool ~~$34.95~~ **$29.95** BW630

OK HOBBY WRAP-30 wire wrap & strip tool **$5.45**

(213) 679-9001

All merchandise sold by California Industrial is premium grade.
Orders are shipped the same day received.
PLEASE INCLUDE $1.00 SHIPPING ON ORDERS UNDER $15.00
California residents add 6% sales tax • Money back guarantee.
Sorry, no COD's • Foreign orders add 10%
Credit cards accepted, $20 minimum





Circle 39 on inquiry card.

What's New?

Binary Synchronous or Synchronous Data Link Control Chip

A synchronous receiver and transmitter chip that can handle either binary synchronous (BiSync) or synchronous data link control (SDLC) protocols in microcomputer systems is available from NEC Microcomputers Inc, Five Militia Dr, Lexington MA 02173. This uPD379 is an N channel MOS device that is packaged in a 42 pin ceramic dual in line package. The part operates at 800 K bps. The uPD379 can operate in full or half duplex mode, is directly TTL compatible, has three state data outputs, has a programmable synchronous word (character), contains detection and rejection of flag, abort and idle patterns, has zero insertion and rejection, and an indication of overrun and underrun errors. The operation mode, data rate and synchronous character of the uPD379 can be changed through the use of external control. The uPD379 is priced at $16 in quantities over 100.■

Circle 627 on inquiry card.

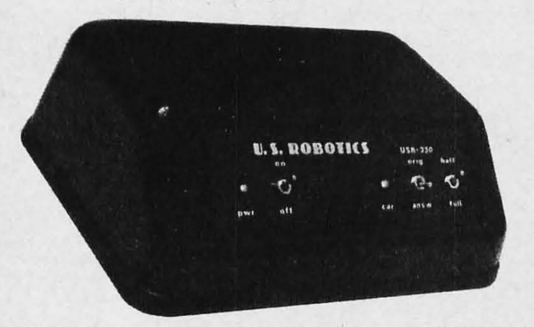
Choice of Speeds in New Acoustic Coupler



The Model AC-312 Acoustic Coupler from Digicom Data Products Inc, 1440 Koll Cir, Suite 108, San Jose CA 95112, offers interchangeable high and low speed capability, field convertible from 300 bps to 1200 bps, using CMOS electronics. The unit accomplishes terminal to computer communication over dial telephone lines. It is Bell 202 half duplex compatible when configured for 1200 bps operation, or Bell 103 compatible at 300 bps configuration. The two in one concept permits users the flexibility of using the same desk styled enclosure for either 300 or 1200 bps operation by field-installing the relevant coupler electronic board. Model AC-312 also has Western Electric compatible 5 bps reverse channel capability. The AC-312 single unit is $495 (1200 bps), Model AC-312 (300 bps) is $245. The Bell compatible 202 electronic board for field upgrading to 1200 bps is $370. The 103 series 300 bps board replacement is $150.■

Circle 631 on inquiry card.

Auto Answer Modem



The USR-320 is a hardwire, asynchronous, auto answer modem that operates in half and full duplex modes at data rates of up to 300 bps. The design uses integrated circuits, crystal controlled digital receiver and transmitter frequencies, and computer designed active filters. The unit comes with power supply and is housed in the desktop case shown. Connection to voice grade telephone lines is via a standard CBS-1001F Data Access Arrangement (DAA). The USR-320 is available with an EIA RS232C interface, a 20 mA current loop interface or both. The USR-330 with RS232C interface is priced at $185; with 20 mA current loop interface $185; and with RS232C and 20 mA current loop interfaces $195. Contact US Robotics Inc, POB 5502, Chicago IL 60680.■

Circle 628 on inquiry card.

Factory Tested Used Modems to Cut Data Communication Costs

The availability of used ICC modems at prices significantly lower than new has been announced by the Special Offer Division of Racal-Milgo Inc, 8600 NW 41st St, Miami FL 33166. According to the company, these modems have been factory tested, and are covered by the same warranty as new equipment. This includes one year for purchased units, and for the full term of units under lease. The used equipment available includes both medium and high speed modems, operating at data rates of 2000 bps, 2400 bps, 3600 bps, 4800 bps, 7200 bps, 9600 bps and 19.2 K bps. Current information on used modems and related equipment is available from the company.■

Circle 630 on inquiry card.

Attention Surplus PDP-8 Owners

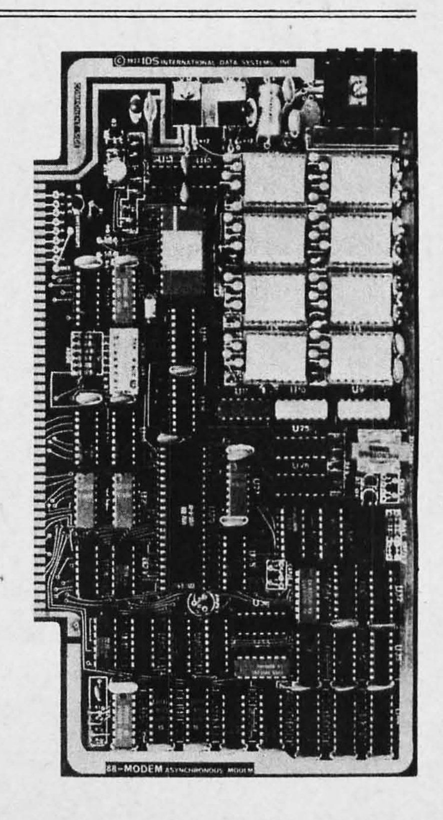
An LED conversion kit for the PDP-8/E and PDP-8/L minicomputers is now available from Scientific Test Systems, POB 741, Wallingford CT 06492. The kits are available to enable replacement of standard incandescent lamps used in the PDP-8/E and PDP-8/L with light emitting diodes, to eliminate the problem of burned out bulbs. The kits are complete with a set of direct replacement LEDs and instructions for modification of the front panel control board circuitry. The conversion kit for the PDP-8/E is priced at $39.95 and $69.95 for the PDP-8/L conversion kit.■

Circle 629 on inquiry card.

New Model 88-Modem

The Model 88-Modem provides communications over either the switched telephone network or private lines at any software selected bps rates between 66 and 600 bps. The modem is fully compatible with Bell System type 103A modems and provides either half or full duplex operation. It is S-100 bus compatible and includes a serial IO port and an originate or answer mode modem on one board. Features implemented in hardware include pulse code dialing in originate mode, automatic break and disconnect, and dial tone detection. The modem includes an 8 pole transmit and 8 pole receiver filter, self-test electronics, dial tone detection, filter as well as error detection electronics including parity, overrun, etc. The dial tone detection circuit allows the dial tone to be positively identified prior to auto dialing of originate calls. Extensive software is included with the 88-Modem including patches for MITS BASIC and North Star DOS version 3. The modem is available in kit or assembled form from International Data Systems Inc, 400 N Washington St, Suite 200, Falls Church VA 22046.■

Circle 632 on inquiry card.



Circle 100 on inquiry card.

Circle 297 on inquiry card.

✱ YOUR BEST BUY IN WIRE WRAP SUPPLIES ✱

PRECUT WIRE

WHY BUY WIRE ON ROLLS?

PRECUT & STRIPPED WIRE IS:

- **Fast** - No more cutting & stripping by hand
- **Reliable** - Good, clean, uniform strip
- **Economical** - Cheaper than using bulk wire

Precut Wire	Bulk Wire
100 pcs of 3" at $.82 = **3¼¢/ft.**	50 ft. roll at $1.99 = **4¢/ft.**
100 pcs of 6" at 1.06 = **2¢/ft.**	100 ft. roll at 2.95 = **3¢/ft.**
Wire Kit 1 at $6.95 = **2 1/3¢/ft.**	

30 Kynar stripped 1" on each end. Lengths are overall
Colors: Red, Blue, Green, Yellow, Black, Orange, White
Wire packaged in plastic bags. Add 25¢/length for tubes.

	100	500	1000	5000
2½ in.	.78	2.40	4.30/K	3.89/K
3 in.	.82	2.60	4.71/K	4.22/K
3½ in.	.86	2.80	5.12/K	4.55/K
4 in.	.90	3.00	5.52/K	4.88/K
4½ in.	.94	3.21	5.93/K	5.21/K
5 in.	.98	3.42	6.34/K	5.52/K
5½ in.	1.02	3.65	6.75/K	5.86/K
6 in.	1.06	3.85	7.16/K	6.19/K
6½ in.	1.15	4.05	7.57/K	6.52/K
7 in.	1.20	4.25	7.98/K	6.85/K
7½ in.	1.25	4.45	8.39/K	7.18/K
8 in.	1.29	4.65	8.80/K	7.53/K
8½ in.	1.32	4.85	9.21/K	7.84/K
9 in.	1.36	5.05	9.62/K	8.17/K
9½ in.	1.40	5.25	10.03/K	8.50/K
10 in.	1.45	5.51	10.44/K	8.83/K
Addl. inches	.10	.41	.82/K	.66/K

WIRE KITS

#1 $6.95

250 3" 100 4½"
250 3½" 100 5"
100 4" 100 6"

#2 $19.95

250 2½" 250 4½" 250 6"
500 3" 250 5" 100 6½"
500 3½" 100 5½" 100 7"
500 4" 1 250 ft Roll Bulk

Choose One Color
or Assortment

SOLDERLESS BREADBOARDS
SK 10 $16.50

2.2" x 6.5"



INCLUDING:
Over 100 pieces of precut wire in assorted lengths — free!
Choose 1 color: Red, Black, Blue, Yellow, Green, White, Orange
Or Assortment.

INTERCONNECT CABLES

Ribbon cable connectors for connecting boards to front panels, or board to board.

	SINGLE ENDED			DOUBLE ENDED		
	14 pin	16 pin	24 pin	14 pin	16 pin	24 pin
6"	1.24	1.34	2.05	2.24	2.45	3.37
12"	1.33	1.44	2.24	2.33	2.55	3.92
24"	1.52	1.65	2.63	2.52	2.76	4.31
48"	1.91	2.06	3.40	2.91	3.17	5.08

WIRE WRAP TOOLS

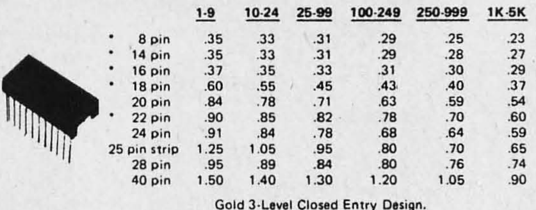


$34.95
With Free Wire Kit 1
($6.95 Value)

HOBBY WRAP
Model BW 630

Batteries & Charger	$11.00
WSU 30 Hand Wrap-Unwrap Strip Tool	6.25
WSU 30M, for Modified Wrap	7.25
BT 30 Extra Bit	2.95

WIRE WRAP SOCKETS



	1-9	10-24	25-99	100-249	250-999	1K-5K
• 8 pin	.35	.33	.31	.29	.25	.23
• 14 pin	.35	.33	.31	.29	.28	.27
• 16 pin	.37	.35	.33	.31	.30	.29
• 18 pin	.60	.55	.45	.43	.40	.37
20 pin	.84	.78	.71	.63	.59	.54
• 22 pin	.90	.85	.82	.78	.70	.60
24 pin	.91	.84	.78	.68	.64	.59
25 pin strip	1.25	1.05	.95	.80	.70	.65
28 pin	.95	.89	.84	.80	.76	.74
40 pin	1.50	1.40	1.30	1.20	1.05	.90

Gold 3-Level Closed Entry Design.
•End & Side Stackable. All prices include Gold.
2 Level Sockets Also Available.

EDGE CARD CONNECTOR SALE!

44 Pin	Solder Tail	$1.75	$15/10
100 Pin	Solder Tail	3.50	$30/10
100 Pin	Wire Wrap	3.50	$30/10

All connectors include Gold.

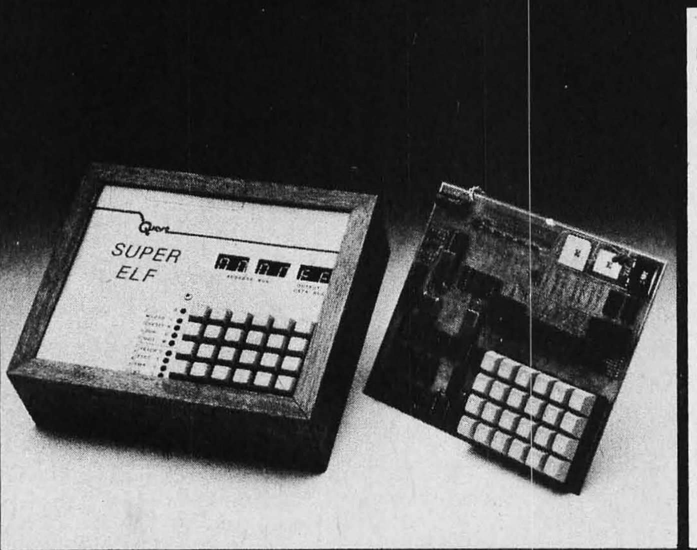
PAGE DIGITAL ELECTRONICS

Ordering Information:

- Orders under $25 and COD's, add $2
- All others, shipped Ppd in U.S. via UPS
- For Blue Label (Air) or 1st Class, add $1
- We accept Visa & Mastercharge
- Most orders shipped same day

135 E. Chestnut Street 5
Monrovia, California 91016
Phone (213) 357-5005

Dealer Inquiries Invited



P.O. Box 4430X Santa Clara, CA 95054
For will call only: (408) 988-1640
2322 Walsh Ave.

Quest ELECTRONICS

Same day shipment. First line parts only. Factory tested. Guaranteed money back. Quality IC's and other components at factory prices.

INTEGRATED CIRCUITS

MICROPROCESSOR	
6800	19.50
8080A with data	8.95
Z80	29.95
8212	2.90
8214	8.00
8216	2.90
8224	2.90
8228	5.35
8251	9.25
8253	10.00
8255	9.25
CDP1802CD	19.95
CDP1802D	25.00
CDP1861	12.95
6820	9.95
6850	12.95
6502	18.50
PROM	
1702A	3.95
N82S23	2.95
N82S123	3.50
N82S126	3.75
N82S129	3.75
N82S131	3.75
N82S136	8.75
N82S137	8.75
2708	12.50
DM8577	2.90
8223	2.90
2716TI	22.50

IC SOCKETS
Solder Tin Low Profile

PIN	1 UP	PIN	1UP
8	.15	24	.35
14	.18	28	.42
16	.20	36	.58
18	.27	40	.61
22	.30		

3 level wire wrap gold.
14 pin .25 16 pin .27
2 level 14 pin ww .21

CONNECTORS	
44 pin edge	2.00
100 pin edge	4.50
100 pin edge WW	5.25

MOS/MEMORY RAM	
2101-1	3.95
2102-1	1.28
2102AL-4	1.60
21F02	1.85
2104A-4	4.95
2107B	4.95
2111-1	4.95
2112-2	3.95
2114	8.50
4116	24.95

2513B	6.30
21L02-1	1.49
MM5262	.40
MM5280	3.00
MM5320	9.95
MM5330	5.94
PD411D-3	4.00
PD411D-4	5.00
P5101	13.95
4200A	12.95
82S25	2.90
91L02A	1.75
HD0165-5	6.95
MM57100	4.50
GIAY38500-1	9.95
MCM6571A	9.95
9368	3.50
INTERFACE	
8T25	3.20
8T26	1.69
8T28	2.75
8T97	1.69
8T98	1.69
8095	.65
8096	.65
8097	.65
8098	.65

CRYSTALS			
1 MHz	4.50	2.0100 MHz	1.95
2 MHz	4.50	2.097152 MHz	4.50
4 MHz	4.25	2.4576 MHz	4.50
5 MHz	4.25	3.2768 MHz	4.50
10 MHz	4.25	5.0688 MHz	4.50
18 MHz	3.90	5.185 MHz	4.50
20 MHz	3.90	5.7143 MHz	4.50
32 MHz	3.90	6.5536 MHz	4.50
32768 Hz	4.00	14.31818 MHz	4.25
1.8432 MHz	4.50	18.432 MHz	4.50
3.5795 MHz	1.20	22.1184 MHz	4.50

60 Hz Crystal Time Base Kit	**$ 4.40**
30 MHz Frequency Counter Kit	**$47.75**
Prescaler Kit to 350 MHz	**$19.95**
COMPUTER BOARD KITS	
8K RAM Board Kit	$134.95
4K EPROM Kit	114.95
I/O Board Kit	44.50
Extender Board w/connector	12.50
Video Interface board kit	125.00
16K EPROM board kit w/o PROMS	74.50
16K Static RAM board kit	395.00
North Star Floppy Disk Kit	**$665.00**
Additional Drive Kit	415.00
Paratronics 100A Logic Analyzer Kit	**$199.00**
Model 10 Trigger Expander Kit	$229.00
Model 150 Bus Grabber Kit	$369.00

New Cosmac Super "ELF"

RCA CMOS expandable to 64K microcomputer w/HEX keypad input and video output for graphics. Just turn on and start loading your program using the resident **monitor on ROM.** Pushbutton selection of all four CPU modes. LED indicators of current CPU mode and four CPU states. Single step op. for program debug. Built in pwr. supply, 256 Bytes of RAM, audio amp. & spkr. Detailed assy. man. w/PC board & all parts fully socketed. Comp. Kit **$106.95.** High address display option **8.95**; Low address display option **9.95**; Custom hardwood cab.; drilled front panel **19.75**; Nicad Battery Backup Kit w/all parts **4.95**; Fully wired & tested in cabinet **151.70**; 1802 software club. 10–12 pg. monthly publication **12.00** per yr.

4K Elf Expansion Board Kit with Cassette I/F $79.95

Available on board options: 1K super ROM monitor **$19.95.** Parallel I/O port **$7.95.** RS232 I/F **$3.50.** TTY 20 ma I/F **$1.95.** S-100 Memory I/F **$4.50.**

Tiny Basic for ANY 1802 System

Cassette **$10.00**
on ROM Monitor **$38.00**
Super Elf owners take 30%off.
Object code listing or paper tape with manual **$5.50**

Auto Clock Kit $15.95

DC clock with 4-.50" displays. Uses National MA-1012 module with alarm option. Includes light dimmer, crystal timebase PC boards. Fully regulated, comp. instructs. Add $3.95 for beautiful dark gray case. Best value anywhere.

'78 IC Update Master Manual

1978 IC Update Master Manual **$30.00**
Complete IC data selector 2175 pg. Master reference guide. Over 42,000 cross references. Free update service through 1978. Domestic postage $3.50. Foreign $6.00. Final 1977 Master closeout **$15.00**

Video Modulator Kit $8.95

Convert your TV set into a high quality monitor without affecting normal usage. Complete kit with full instructions.

RCA CosmacVIP Kit $275.00

Video computer with games and graphics.

Sinclair 3½ Digit Multimeter $59.95

Batt. oper. 1mV and .1NA resolution. Resistance to 20 meg. 1% accuracy. Small, portable, completely assem. in case. 1 yr. guarantee.

PROM Eraser

Ultraviolet, assembled **$49.95**

TERMS: $5.00 min. order U.S. Funds. Calif residents add 6% tax. BankAmericard and Master Charge accepted. Shipping charges will be added on charge cards.
FREE: Send for your copy of our NEW 1978 QUEST CATALOG. Include 28¢ stamp.

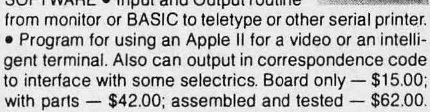
Circle 307 on inquiry card.

SAVE COMPUTER INTERFACES & PERIPHERALS SAVE

For free catalog including parts lists and schematics, send a self-addressed stamped envelope.

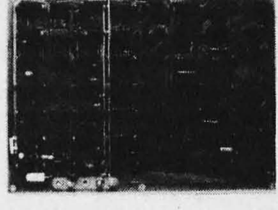
APPLE II SERIAL I/O INTERFACE *

Part no. 2
Baud rate is continuously adjustable from 0 to 30,000 • Plugs into any peripheral connector • Low current drain. RS-232 input and output • On board switch selectable 5 to 8 data bits, 1 or 2 stop bits, and parity or no parity either odd or even • Jumper selectable address • SOFTWARE • Input and Output routine from monitor or BASIC to teletype or other serial printer. • Program for using an Apple II for a video or an intelligent terminal. Also can output in correspondence code to interface with some selectrics. Board only — $15.00; with parts — $42.00; assembled and tested — $62.00.



T.V. TYPEWRITER





Part no. 106
• Stand alone TVT • 32 char/line, 16 lines, modifications for 64 char/line included • Parallel ASCII (TTL) input • Video output • 1K on board memory • Output for computer controlled curser • Auto scroll • Non-destructive curser • Curser inputs: up, down, left, right, home, EOL, EOS • Scroll up, down • Requires +5 volts at 1.5 amps, and -12 volts at 30 mA • All 7400, TTL chips • Char. gen. 2513 • Upper case only • Board only $39.00; with parts $145.00

8K STATIC RAM

Part no. 300
• 8K Altair bus memory • Uses 2102 Static memory chips • Memory protect • Gold contacts • Wait states • On board regulator • S-100 bus compatible • Vector input option • TRI state buffered • Board only $22.50; with parts $160.00

RF MODULATOR *





Part no. 107
• Converts video to AM modulated RF, Channels 2 or 3. So powerful almost no tuning is required. On board regulated power supply makes this extremely stable. Rated very highly in Doctor Dobbs' Journal. Recommended by Apple. • Power required is 12 volts AC C.T., or +5 volts DC • Board $7.60; with parts $13.50

MODEM *

Part no. 109
• Type 103 • Full or half duplex • Works up to 300 baud • Originate or Answer • No coils, only low cost components • TTL input and output-serial • Connect 8 ohm speaker and crystal mic. directly to board • Uses XR FSK demodulator • Requires +5 volts • Board $7.60; with parts $27.50

TIDMA *

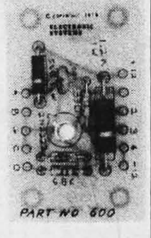
Part no. 112
• Tape Interface Direct Memory Access • Record and play programs without bootstrap loader (no prom) has FSK encoder/decoder for direct connections to low cost recorder at 1200 baud rate, and direct connections for inputs and outputs to a digital recorder at any baud rate. • S-100 bus compatible • Board only $35.00; with parts $110.00

DC POWER SUPPLY *



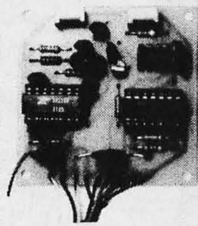
Part no. 6085
• Board supplies a regulated +5 volts at 3 amps., +12, -12, and -5 volts at 1 amp. • Power required is 8 volts AC at 3 amps., and 24 volts AC C.T. at 1.5 amps. • Board only $12.50; with parts excluding transformers $42.50

RS 232/TTY * INTERFACE



Part no. 600
• Converts RS-232 to 20mA current loop, and 20mA current loop to RS-232 • Two separate circuits • Requires +12 and -12 volts • Board only $4.50, with parts $7.00

TAPE INTERFACE *

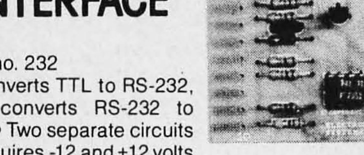


Part no. 111
• Play and record Kansas City Standard tapes • Converts a low cost tape recorder to a digital recorder • Works up to 1200 baud • Digital in and out are TTL-serial • Output of board connects to mic. in of recorder • Earphone of recorder connects to input on board • No coils • Requires +5 volts, low power drain • Board $7.60; with parts $27.50

UART & BAUD RATE GENERATOR *

Part no. 101
• Converts serial to parallel and parallel to serial • Low cost on board baud rate generator • Baud rates: 110, 150, 300, 600, 1200, and 2400 • Low power drain +5 volts and -12 volts required • TTL compatible • All characters contain a start bit, 5 to 8 data bits, 1 or 2 stop bits, and either odd or even parity. • All connections go to a 44 pin gold plated edge connector • Board only $12.00; with parts $35.00 with connector add $3.00

RS 232/TTL* INTERFACE



Part no. 232
• Converts TTL to RS-232, and converts RS-232 to TTL • Two separate circuits • Requires -12 and +12 volts • All connections go to a 10 pin gold plated edge connector • Board only $4.50; with parts $7.00 with connector add $2.00

ELECTRONIC SYSTEMS Dept. B, P.O. Box 21638, San Jose, CA. USA 95157

To Order: Mention part number and description. For parts kits add "A" to part number. In USA, shipping paid for orders accompanied by check, money order, or Master Charge, BankAmericard, or VISA number, expiration date and signature. Shipping charges added to C.O.D. orders. California residents add 6.5% for tax. Outside USA add 10% for air mail postage, no C.O.D.'s. Checks and money orders must be payable in US dollars. Parts kits include sockets for all ICs, components, and circuit board. Documentation is included with all products. All items are in stock, and will be shipped the day order is received via first class mail. Prices are in US dollars. No open accounts. To eliminate tariff in Canada boxes are marked "Computer Parts." Dealer inquiries invited.
24 Hour Order Line: (408) 226-4064

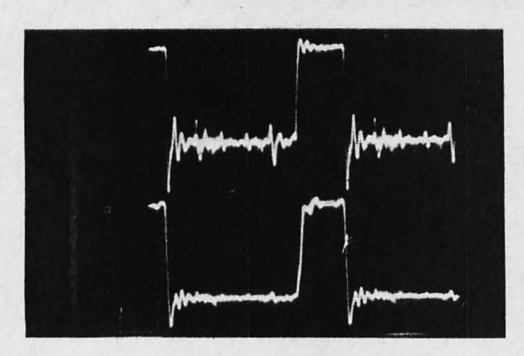




✱ Circuits designed by John Bell

What's New?

Glitch Grabber Reduces Noise, Glitches, Jitter on S-100 Bus



A board interconnection device that reduces noise, glitches and jitter on the S-100 microcomputer bus has been announced by Extensys Corp, 380 Bernardo Av, Mountain View CA 94040. Called the Extensys Glitch Grabber, the board helps to clean signals on the S-100 bus. The device provides glitch-free signals by applying analog techniques from transmission line analysis. It plugs into any open slot on the S-100 bus and features a proprietary self-regulating transistor network that controls voltages. The electronics are activated only when the glitch is there, to minimize any effects on S-100 bus signals. The price is $79.50.■

Circle 547 on inquiry card.

Attention Chess Nuts

The MACC-1 is a new companion to the Chess Challenger and is designed for advanced players. The unit will never play the same game twice and will not allow illegal moves including accidental moving into check. It does allow castling and pawn capture en passant. The programming concentrates heavily on improved end of game tactics and will not miss checkmate situations. Multiple levels of difficulty can be selected or changed at any time throughout the game. Chess problems may be set up quickly by telling the computer what pieces should be positioned, where they are to be placed, and the game starts from there. Any previous Chess Challenger may be upgraded to the MACC-1. This new unit is priced at $250. Contact Fidelity Electronics Ltd, 5245 Diversey Av, Chicago IL 60639.■

Circle 548 on inquiry card.

NEW ANNOUNCING

The EW-2001 A "Smart" VIDEO BOARD KIT At A "Dumb" Price!

A VIDEO BOARD + A MEMORY BOARD + AN I/O BOARD – ALL IN ONE!

■ STATE OF THE ART TECHNOLOGY USING DEDICATED MICROPROCESSOR I.C. ■ NUMBER OF I.C.s REDUCED BY 50% FOR HIGHER RELIABILITY ■ MASTER PIECE OF ENGINEERING ■ FULLY SOFTWARE CONTROLLED

Priced at ONLY **$199.95** Basic Software Included

SPECIAL FEATURES:

- S-100 bus compatible
- Parallel keyboard port
- On board 4K screen memory (optional)* relocatable to main computer memory
- Text editing capabilities (software optional)
- Scrolling: up and down through video memory
- Blinking characters
- Reversed video
- Provision for on board ROM
- CRT and video controls fully programmable (European TV)
- Programmable no. of scan lines
- Underline blinking cursor
- Cursor controls: up, down, left, right, home, carriage return
- Composite video

*Min. 2K required for operation of this board.

DISPLAY FEATURES:

- 128 displayable ASCII characters (upper and lower case alphanumeric, controls)
- 64 or 32 characters per line (jumper selectable)
- 32 or 16 lines (jumper selectable)
- Screen capacity 2048 or 512
- Character generation: 7 x 11 dot matrix

OPTIONS:

Sockets	$10.00
2K Static Memory (with Sockets)	$45.00
4K Static Memory (with Sockets)	$90.00
Complete unit, assembled and tested with 4K Memory	$335.00
Basic software on ROM	$20.00
Text editor on ROM	$75.00

DEALER INQUIRIES WELCOMED

8080 SUPPORT

8212	$3.00
8214	7.95
8216	3.50
8228	5.95
8251	7.95
8555	8.50

8080A CPU $7.75

RAM – 2114 1Kx4 450ns $8.00

1/4W RESISTOR
10 Ohm – 1.5m
$1.75/100
of one value

RIBBON CABLE
32 Conductor
26 AWG – $.60/Foot

1 Pole 10 Pos. ROTARY SWITCH 3 for $1.00

MINIATURE Slide Switch DPDT $.15; 10/$1.00

1 Pole 8 Pos. TO5 Miniature Rotary Switch 3 for $1.00

Push Button Momentary Switch 3 for $1.00

HEXADECIMAL LABEL KEYBOARD

- Matrix coded output
- Interfaces with 74C922 for binary code
- Zero bounce
- Est. life: 100 million
- Remove back to stick on

SHIPPING: Keyboard and Video Board: $3.50; others: $1.25

California residents add 6% sales tax

ELECTRONICS WAREHOUSE Inc.

1603 AVIATION BLVD.
REDONDO BEACH, CA. 90278
TEL. (213) 376-8005



master charge



WRITE FOR FREE CATALOG

Minimum Order: $10

ASCII 3rd GENERATION KEYBOARD KIT *ONLY $68.00

- TTL Logic Circuits
- Power: +5V 275mA
- Upper and Lower Case
- Full ASCII Set (Alpha Numeric, Symbols, Control)
- 7 or 8 Bits Parallel Data
- Optional Serial Output
- Selectable Positive or Negative Strobe, and Strobe Pulse Width
- 'N' Key Roll-Over
- Fully Debounced
- Carriage Return Key
- Repeat Function Key
- Shift Lock, 2 Shift Keys
- 4 User Defineable Keys
- P.C. Board Size: 17-3/16" x 5"

OPTIONS:

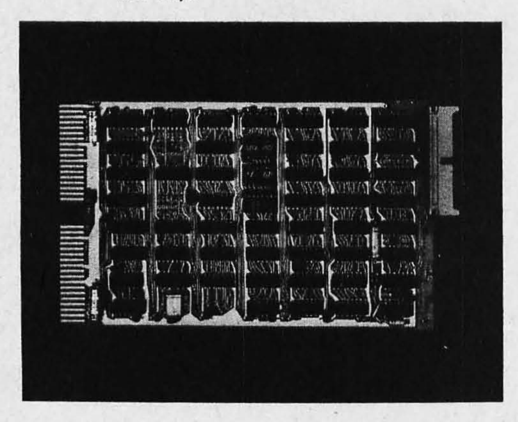
Metal Enclosure Painted IBM Blue and White)	$25.00
18 Pin Edge Con.	$2.00
I.C. Sockets	$4.00
Serial Output (Shift Register)	$2.00
Upper Case Lock Switch for Capital Letters and Numbers	$2.00

KIT INCLUDES: Keyboard, P.C. Board, all required components & assembly manual.

NOTE: If you have this 63 Key Teletype Keyboard you can buy the Kit without it for only $44.95.

Circle 130 on inquiry card.

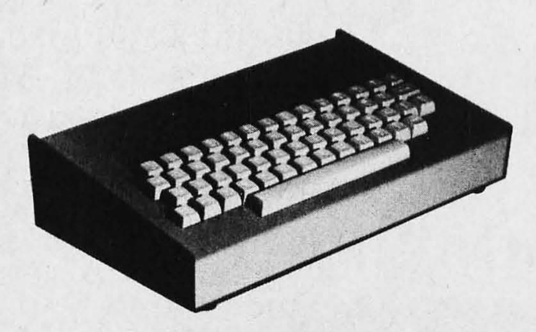
DMA Interface for LSI-11 and LSI-11/2



This general purpose DMA interface card is now available from Computer Technology, 6043 Lawton Av, Oakland CA 94618. The card features the following: dual height card (8.5 by 5 inches or 22 by 13 cm); bootstrap facility provided for the user's programmable read only memory; ease of interfacing to the user's device by means of a buffered bidirectional data bus; allows direct programmed IO with five 16 bit read and write registers in the user's device; handles byte or word transfers at rates up to 400 K transfers per second; handles extended addressing up to 128 K words; burst mode capability; useful with floppy or hard disk controllers, line printers, interprocessor communication, data acquisition and many other high performance applications. Priced at $495 from Computer Technology.■

Circle 584 on inquiry card.

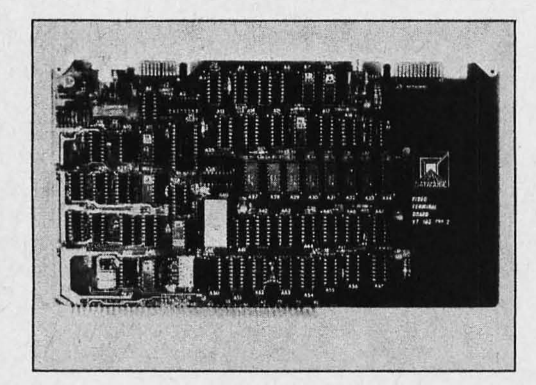
ASCII Keyboard in Kit or Assembled



The Model 756 ASCII keyboard provides encoding for all 128 ASCII characters and control functions. Utilizing KBM Series keyswitches and low power MOS encoder circuitry, Model 756 is designed to bridge the gap between stripped down basic keyboards and custom models. Accessories include a numeric pad, custom cables and connectors. The new 702 enclosure features all steel construction and is supplied in a wrinkle finish to match modern hardware design. The interface allows user selection of parity, positive or negative logic data and strobe outputs, alpha lock operation and both DC level and pulse strobe signals. A latching shift lock key is included and all outputs are TTL DTL MOS compatible. The 756 carries a 90 day warranty. The price for the Model 756 kit is $64.95. The assembled and tested model retails for $75.95. The matching enclosure, Model 702, is $29.95. Contact George Risk Industries Inc, GRI Plaza, Kimball NB 69145.■

Circle 585 on inquiry card.

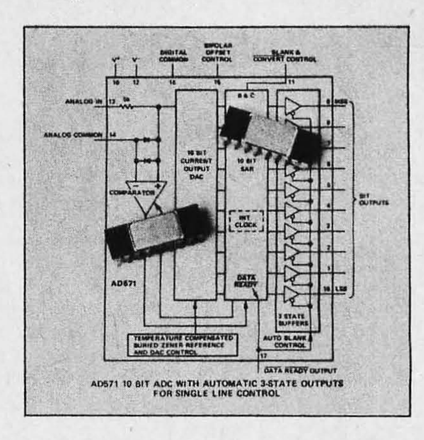
Video Terminal Board for Intel and National Computers



The Datacube VT 103 Video Terminal Board interfaces directly to the system bus of the Intel SBC 80 Series and National BLC 80 Series. The board provides a 96 character ASCII subset in seven by nine font in a 64 character by 16 line format as a video signal to drive an external monitor. The board is addressed as an IO device and the device code programming is switch selectable. The unit provides direct cursor addressing and 11 other cursor control functions. The Datacube VT 103 has an input port for an optional external keyboard. Use of the keyboard can replace a Teletype or other terminals in many applications. Inputs for a strobe and seven data lines are provided at a 26 pin PC edge connector along with a +5 and −12 V from the system bus for keyboard power. The composite video can drive a 75 ohm coaxial cable with a 1.4 V peak to peak signal and meets RS420 standards. The board sells for $275 in lots of 100 and can be obtained from Datacube/SMK-1, 670 Main St, Reading MA 01867.■

Circle 586 on inquiry card.

Complete Monolithic 10 Bit Analog to Digital Converter



AD571 10 BIT ADC WITH AUTOMATIC 3-STATE OUTPUTS FOR SINGLE LINE CONTROL

This complete, 10 bit monolithic analog to digital converter combines linear and digital circuitry on a single integrated circuit chip. The AD571 is produced using the integrated-injection logic (I^2L) technique which allows very high circuit densities to be fabricated on a single chip. The AD571 is also the first monolithic analog to digital converter to be laser wafer trimmed. The converter uses the successive approximation technique and includes a DAC, voltage reference, clock, comparator, successive approximation register and output buffer on a 120 by 150 mil (30 mm by 38 mm) chip. The device executes a complete conversion to 10 bit accuracy ±1/2 lsb in 25 μs over the specified temperature range. The AD571 is available in three versions. For fully guaranteed 10 bit performance at 25° the least expensive is the J version. The K version provides 10 bit accuracy over the commerical temperature range of 0 thru 70° C. If you need a −55 to +125 °C temperature range, the S version is your choice. AD571 is priced in 100s at $24, $35 and $60 for the J, K and S respectively. Contact Analog Devices Semiconductor, 829 Woburn St, Wilmington MA 01887.■

Circle 587 on inquiry card.

Where Do New Product Items Come From?

The information printed in the new products pages of BYTE is obtained from "new product" or "press release" copy sent by the promoters of new products. If in our judgment the neat new whizbang gizmo or save the world software package is of interest to the personal computing experimenters and homebrewers who read BYTE, we print the information in some form. We openly solicit such information from manufacturers and suppliers to this marketplace. The information is printed more or less as a first in first out queue, subject to occasional priority modifications.

DIODES/ZENERS

1N914	100v	10mA	.05
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.05
1N4733	5.1v	1 W Zener	.25
1N753A	6.2v	500 mW Zener	.25
1N758A	10v	"	.25
1N759A	12v	"	.25
1N5243	13v	"	.25
1N5244B	14v	"	.25
1N5245B	15v	"	.25

SOCKETS/BRIDGES

8-pin	pcb	.20	ww	.35
14-pin	pcb	.20	ww	.40
16-pin	pcb	.20	ww	.40
18-pin	pcb	.25	ww	.75
22-pin	pcb	.35	ww	.95
24-pin	pcb	.35	ww	.95
28-pin	pcb	.45	ww	1.25
40-pin	pcb	.50	ww	1.25

Molex pins	.01	To-3 Sockets	.25
2 Amp Bridge		100-prv	.95
25 Amp Bridge		200-prv	1.95

TRANSISTORS, LEDS, etc.

2N2222	NPN	(2N2222 Plastic .10)	.15
2N2907	PNP		.15
2N3906	PNP	(Plastic - Unmarked)	.10
2N3904	NPN	(Plastic - Unmarked)	.10
2N3054	NPN		.35
2N3055	NPN	15A 60v	.50
T1P125	PNP	Darlington	.35
LED Green, Red, Clear, Yellow			.15
D.L.747	7 seg 5/8" High com-anode		1.95
MAN72	7 seg com-anode (Red)		1.25
MAN3610	7 seg com-anode (Orange)		1.25
MAN82A	7 seg com-anode (Yellow)		1.25
MAN74A	7 seg com-cathode (Red)		1.50
FND359	7 seg com-cathode (Red)		1.25

C MOS

4000	.15
4001	.15
4002	.20
4004	3.95
4006	.95
4007	.20
4008	.75
4009	.35
4010	.35
4011	.20
4012	.20
4013	.40
4014	.75
4015	.75
4016	.35
4017	.75
4018	.75
4019	.35
4020	.85
4021	.75
4022	.75
4023	.20
4024	.75
4025	.20
4026	1.95
4027	.35
4028	.75
4030	.35
4033	1.50
4034	2.45
4035	.75
4040	.75
4041	.69
4042	.65
4043	.50
4044	.65
4046	1.25
4049	.45
4050	.45
4066	.55
4069/74C04	.25
4071	.25
4081	.30
4082	.30
MC 14409	14.50
MC 14419	4.85
4511	.95
74C151	1.90

– T T L –

7400	.10	7473	.25	74176	.85	74H72	.35	74S133	.40
7401	.15	7474	.30	74180	.55	74H101	.75	74S140	.55
7402	.15	7475	.35	74181	2.25	74H103	.55	74S151	.30
7403	.15	7476	.40	74182	.75	74H106	.95	74S153	.35
7404	.10	7480	.55	74190	1.25			74S157	.75
7405	.25	7481	.75	74191	.95	74L00	.25	74S158	.30
7406	.25	7483	.75	74192	.75	74L02	.20	74S194	1.05
7407	.55	7485	.55	74193	.85	74L03	.25	74S257 (8123)	1.05
7408	.15	7486	.25	74194	.95	74L04	.30		
7409	.15	7489	1.05	74195	.95	74L10	.20	74LS00	.20
7410	.15	7490	.45	74196	.95	74L20	.35	74LS01	.20
7411	.25	7491	.70	74197	.95	74L30	.45	74LS02	.20
7412	.25	7492	.45	74198	1.45	74L47	1.95	74LS04	.20
7413	.25	7493	.35	74221	1.00	74L51	.45	74LS05	.25
7414	.75	7494	.75	74367	.75	74L55	.65	74LS08	.25
7416	.25	7495	.60			74L72	.45	74LS09	.25
7417	.40	7496	.80	75108A	.35	74L73	.40	74LS10	.25
7420	.15	74100	1.15	75491	.50	74L74	.45	74LS11	.25
7426	.25	74107	.25	75492	.50	74L75	.55	74LS20	.20
7427	.25	74121	.35			74L93	.55	74LS21	.25
7430	.15	74122	.55			74L123	.85	74LS22	.25
7432	.20	74123	.35	74H00	.15			74LS32	.25
7437	.20	74125	.45	74H01	.20	74S00	.35	74LS37	.25
7438	.20	74126	.35	74H04	.20	74S02	.35	74LS38	.35
7440	.20	74132	.75	74H05	.20	74S03	.25	74LS40	.30
7441	1.15	74141	.90	74H08	.35	74S04	.25	74LS42	.65
7442	.45	74150	.85	74H10	.35	74S05	.35	74LS51	.35
7443	.45	74151	.65	74H11	.25	74S08	.35	74LS74	.35
7444	.45	74153	.75	74H15	.45	74S10	.35	74LS86	.35
7445	.65	74154	.95	74H20	.25	74S11	.35	74LS90	.55
7446	.70	74156	.70	74H21	.25	74S20	.25	74LS93	.55
7447	.70	74157	.65	74H22	.40	74S40	.20	74LS107	.40
7448	.50	74161	.55	74H30	.20	74S50	.20	74LS123	1.00
7450	.25	74163	.85	74H40	.25	74S51	.25	74LS151	.75
7451	.25	74164	.60	74H50	.25	74S64	.15	74LS153	.75
7453	.20	74165	1.10	74H51	.25	74S74	.35	74LS157	.75
7454	.25	74166	1.25	74H52	.15	74S112	.60	74LS164	1.00
7460	.40	74175	.80	74H53J	.25	74S114	.65	74LS193	.95
7470	.45			74H55	.20			74LS367	.75
7472	.40							74LS368	.65

LINEARS, REGULATORS, etc.

MCT2	.95					LM723	.40
8038	3.95	LM320T5	1.65	LM340K15	1.25	LM725N	2.50
LM201	.75	LM320T12	1.65	LM340K18	1.25	LM739	1.50
LM301	.45	LM320T15	1.65	LM340K24	1.25	LM741 (8-14)	.25
LM308 (Mini)	.95	LM324N	1.25	78L05	.75	LM747	1.10
LM309H	.65	LM339	.75	78L12	.75	LM1307	1.25
LM309K (340K-5)	.85	7805 (340T5)	.95	78L15	.75	LM1458	.65
LM310	.85	LM340T12	.95	78M05	.75	LM3900	.50
LM311D (Mini)	.75	LM340T15	.95	LM373	2.95	LM75451	.65
LM318 (Mini)	1.75	LM340T18	.95	LM380 (8-14 PIN)	.95	NE555	.35
LM320K5 (7905)	1.65	LM340T24	.95	LM709 (8, 14 PIN)	.25	NE556	.85
LM320K12	1.65	LM340K12	1.25	LM711	.45	NE565	.95
						NE566	1.25
						NE567	.95

9000 SERIES

9301	.85	95H03	1.10
9309	.35	9601	.20
9322	.65	9602	.45

MICRO'S, RAMS, CPU'S, E-PROMS

74S188	3.00	8214	8.95
1702A	4.50	8224	3.25
MM5314	3.00	8228	6.00
MM5316	3.50	8251	8.50
2102-1	1.45	8255	10.50
2102L-1	1.75	8T13	1.50
2114	9.50	8T23	1.50
TR1602B	3.95	8T24	2.00
TMS 4044-	9.95	8T97	1.00
		2107B-4	4.95
8080	8.95	2708	9.50
8212	2.95	Z80 PIO	8.50

INTEGRATED CIRCUITS UNLIMITED

7889 Clairemont Mesa Boulevard, San Diego, California 92111
(714) 278-4394 *(Calif. Res.)*

All orders shipped prepaid — No minimum
Open accounts invited — COD orders accepted
Discounts available at OEM Quantities — California Residents add 6% Sales Tax
All IC's Prime/Guaranteed. All orders shipped same day received.

24 Hour Toll Free Phone 1-800-854-2211

American Express / BankAmericard / Visa / MasterCharge

SPECIAL DISCOUNTS

Total Order	Deduct
$35 - $99	10%
$100 - $300	15%
$301 - $1000	20%

Circle 180 on inquiry card.

Circle 200 on inquiry card.

7400 TTL

SN7400N	.16				
SN7401N	.18	SN7472N	.29	SN74160N	.89
SN7402N	.18	SN7473N	.35	SN74161N	.89
SN7403N	.18	SN7474N	.35	SN74162N	1.95
SN7404N	.18	SN7475N	.49	SN74163N	.89
SN7405N	.20	SN7476N	.35	SN74164N	.89
SN7406N	.29	SN7479N	5.00	SN74165N	.89
SN7407N	.29	SN7480N	.50	SN74166N	1.25
SN7408N	.20	SN7482N	.99	SN74167N	1.95
SN7409N	.20	SN7483N	.59	SN74170N	1.59
SN7410N	.18	SN7485N	.79	SN74172N	6.00
SN7411N	.25	SN7486N	.35	SN74173N	1.25
SN7412N	.25	SN7489N	1.75	SN74174N	.89
SN7413N	.40	SN7490N	.45	SN74175N	.79
SN7414N	.70	SN7491N	.59	SN74176N	.79
SN7416N	.25	SN7492N	.43	SN74177N	.79
SN7417N	.25	SN7493N	.43	SN74179N	1.95
SN7420N	.20	SN7494N	.65	SN74180N	.79
SN7421N	.29	SN7495N	.65	SN74181N	1.95
SN7422N	.39	SN7496N	.65	SN74182N	.79
SN7423N	.25	SN7497N	3.00	SN74184N	1.95
SN7425N	.29	SN74100N	.89	SN74185N	1.95
SN7426N	.29	SN74107N	.35	SN74186N	9.95
SN7427N	.25	SN74109N	.59	SN74188N	3.95
SN7429N	.39	SN74116N	1.95	SN74190N	1.25
SN7430N	.20	SN74121N	.35	SN74191N	1.25
SN7432N	.25	SN74122N	.39	SN74192N	.79
SN7437N	.25	SN74123N	.49	SN74193N	.79
SN7438N	.25	SN74125N	.49	SN74194N	.89
SN7439N	.25	SN74126N	.49	SN74195N	.69
SN7440N	.20	SN74132N	.75	SN74196N	.89
SN7441N	.89	SN74136N	.75	SN74197N	.89
SN7442N	.49	SN74141N	.79	SN74198N	1.49
SN7443N	.75	SN74142N	2.95	SN74199N	1.49
SN7444N	.75	SN74143N	2.95	SN74200N	5.59
SN7445N	.75	SN74144N	2.95	SN74251N	1.79
SN7446N	.69	SN74145N	.79	SN74279N	.79
SN7447N	.59	SN74147N	1.95	SN74283N	2.25
SN7448N	.79	SN74148N	1.29	SN74284N	3.95
SN7450N	.20	SN74150N	.89	SN74285N	3.95
SN7451N	.20	SN74151N	.59	SN74365N	.69
SN7453N	.20	SN74153N	.59	SN74366N	.69
SN7454N	.20	SN74154N	.99	SN74367N	.69
SN7459A	.25	SN74155N	.79	SN74368N	.69
SN7460N	.20	SN74156N	.79	SN74390N	1.95
SN7470N	.29	SN74157N	.65	SN74393N	1.95

C/MOS

CD4000	.23			CD4070	.55
CD4001	.23			CD4071	.23
CD4002	.23	CD4028	.89	CD4072	.49
CD4006	1.19	CD4029	1.19	CD4076	1.39
CD4007	.25	CD4030	.49	CD4081	.23
CD4009	.49	CD4035	.99	CD4082	.23
CD4010	.49	CD4040	1.19	CD4093	.99
CD4011	.23	CD4041	1.25	CD4098	2.49
CD4012	.25	CD4042	.99	MC14409	14.95
CD4013	.39	CD4043	.89	MC14410	14.95
CD4014	1.39	CD4044	.89	MC14411	14.95
CD4015	1.19	CD4046	1.79	MC14419	4.95
CD4016	.49	CD4047	2.50	MC14433	19.95
CD4017	1.19	CD4048	1.35	MC14506	.75
CD4018	.99	CD4049	.49	MC14507	.99
CD4019	.49	CD4050	.49	MC14562	14.50
CD4020	1.19	CD4051	1.19	MC14583	3.50
CD4021	1.39	CD4053	1.19	CD4508	3.95
CD4022	1.19	CD4056	2.95	CD4510	1.39
CD4023	.23	CD4059	9.95	CD4511	1.29
CD4024	.79	CD4060	1.49	CD4515	2.95
CD4025	.23	CD4066	.79	CD4518	1.29
CD4026	2.25	CD4068	.39	CD4520	1.29
CD4027	.69	CD4069	.45	CD4566	2.25

74C00

74C00	.39			74C163	3.00
74C02	.55			74C164	3.25
74C04	.75	74C89	6.49	74C173	2.60
74C08	.75	74C90	3.00	74C192	3.49
74C10	.65	74C93	2.00	74C193	2.75
74C11	3.00	74C95	2.00	74C195	2.75
74C20	.65	74C107	1.25	74C922	9.95
74C30	.65	74C151	2.90	74C923	8.95
74C42	2.15	74C154	3.00	74C925	14.95
74C48	4.75	74C157	2.15	74C926	11.95
74C73	1.50	74C160	3.25	80C95	1.50
74C74	1.15	74C161	3.25	80C97	1.50

LINEAR

78MG	1.75			LM733N	1.00
LM300H	.80			LM739N	1.19
LM301H	.35			LM741CH	.35
LM301CN	.35			LM741CN	.35
LM302H	.75	LM340T-8	1.25	LM741-14N	.39
LM304H	1.00	LM340T-12	1.25	LM747H	.79
LM305H	.60	LM340T-15	1.25	LM747N	.79
LM307CN/H	.35	LM340T-18	1.25	LM748H	.39
LM308H	1.00	LM340T-24	1.25	LM748N	.39
LM308CN	1.00	LM350N	1.00	LM1303N	.90
LM309H	1.10	LM351CN	.65	LM1304N	1.19
LM309K	1.25	LM370N	1.15	LM1305N	1.40
LM310CN	1.15	LM373N	3.25	LM1307N	.85
LM311H	.90	LM377N	4.00	LM1310N	2.95
LM311N	.90	LM380N	1.25	LM1351N	1.65
LM317K	6.50	LM380CN	.99	LM1414N	1.75
LM318CN	1.50	LM381N	1.79	LM1458CN/H	.59
LM319N	1.30	LM382N	1.79	MC1488	1.95
LM320K-5	1.35	NE501N	8.00	MC1489	1.95
LM320K-5.2	1.35	NE510A	6.00	LM1496N	.95
LM320K-12	1.35	NE529A	4.95	LM1556V	1.75
LM320K-15	1.35	NE531H	3.00	MC1741SCP	3.00
LM320K-18	1.35	NE536T	6.00	LM2901N	2.95
LM320K-24	1.35	NE540L	6.00	LM3053	1.50
LM320T-5	1.25	NE550N	1.30	LM3065N	.69
LM320T-5.2	1.25	NE555V	.39	LM3900N(3401)	.49
LM320T-8	1.25	NE556	.99	LM3905N	.89
LM320T-12	1.25	NE560B	5.00	LM3909	1.25
LM320T-15	1.25	NE561B	5.00	MC5558V	1.00
LM320T-18	1.25	NE562B	5.00	LM7525N	.90
LM320T-24	1.25	NE565H	1.75	LM7534N	.75
LM323K-5	5.95	NE565N	1.25	8038B	4.95
LM324N	1.80	NE566CN	1.75	LM75450	.49
LM339N	.99	NE567H	1.25	75451CN	.39
LM340K-5	1.35	NE567V	.99	75452CN	.39
LM340K-6	1.35	NE570	10.50	75453CN	.39
LM340K-8	1.35	LM703CN/H	.45	75454CN	.39
LM340K-12	1.35	LM709H	.29	75491CN	.79
LM340K-15	1.35	LM709N	.29	75492CN	.89
LM340K-18	1.35	LM710N	.79	75494CN	.89
LM340K-24	1.35	LM711N	.39	RC4151	5.95
LM340T-5	1.25	LM723H	.55	RC4194	5.95
LM340T-6	1.25	LM723N	.55	RC4195	4.49

74LS00 TTL

74LS00	.23			74LS139	.69
74LS01	.23			74LS151	.69
74LS02	.23	74LS51	.23	74LS155	.69
74LS03	.23	74LS54	.23	74LS157	.69
74LS04	.29	74LS55	.23	74LS160	.89
74LS05	.29	74LS73	.35	74LS161	.89
74LS08	.23	74LS74	.35	74LS162	.89
74LS09	.29	74LS75	.49	74LS163	.89
74LS10	.23	74LS76	.35	74LS164	.99
74LS13	.49	74LS83	.75	74LS175	.79
74LS14	.99	74LS85	.99	74LS181	2.49
74LS15	.29	74LS86	.35	74LS190	.89
74LS20	.23	74LS90	.49	74LS191	.89
74LS21	.29	74LS92	.59	74LS192	.89
74LS22	.29	74LS93	.59	74LS193	.89
74LS26	.29	74LS95	.79	74LS194	.89
74LS27	.29	74LS96	.89	74LS195	.89
74LS28	.29	74LS107	.35	74LS253	.79
74LS30	.23	74LS109	.35	74LS257	.69
74LS32	.29	74LS112	.35	74LS260	.55
74LS37	.35	74LS123	.99	74LS279	.59
74LS40	.29	74LS132	.79	74LS367	.59
74LS42	.69	74LS136	.39	74LS368	.59
74LS47	.69	74LS138	.69	74LS670	1.95

BUGBOOK®

Continuing Education Series

BUGBOOKS I and II — $17.00 per set
by Peter R. Rony, David G. Larsen, WB4HYJ
Sold as a set these two books outline over 90 experiments designed to teach the reader all he will need to know about TTL logic chips to use them in conjunction with microprocessor systems. You'll learn about the basic concepts of digital electronics including gates, flip-flops, latches, buses, decoders, multiplexers, demultiplexers, LED displays, RAM's, ROM's, and much, much more.

BUGBOOK IIa — $5.00
by Peter R. Rony, David G. Larsen, WB4HYJ
This volume will introduce you to the fabulous UART chip — that all important interface between data terminals, etc., and your microcomputer. It also covers current loops, and the RS 232C interface standard. Particularly recommended for any RTTY enthusiast.

BUGBOOK III — $15.00
by Peter R. Rony, David G. Larsen, WB4HYJ, Jonathan A. Titus
Here is the book that puts it all together. Besides having much valuable text there are a series of experiments in which the reader completely explores the 8080 chip pin by pin and introduces you to the Mark 80 microcomputer, a unique easily interfaced system. It is recommended that you have the background on the BUGBOOKS I & II before proceeding with BUGBOOK III.

BUGBOOK V and VI — $19.00 per set
by David G. Larsen, Peter R. Rony, Jonathan A. Titus
Experiments in digital electronics, 8080A microcomputer programming and 8080A microcomputer interfacing. An integrated approach to self-instructed basic digital electronics, breadboarding and 8080A interfacing/programming. Bugbook VI integrates the digital concepts of Bugbook V into a treatment of 8080A microcomputer programming and interfacing. Detail & laboratory experiments included with each book.

CMOS-M — DESIGNERS PRIMER AND HANDBOOK New expanded version — $8.50
Starts at basic structure of CMOS devices through integration into MSI.

THE 555 TIMER APPLICATIONS SOURCEBOOK WITH EXPERIMENTS — $6.95
by Howard M. Berlin W3HB
This book shows you what the 555 timer is and how to use it. Included are over 100 various design techniques, equations and graphs to create "ready-to-go" timers, generators, power supplies, measurement and control circuits, party games, circuits for the home and automobile, photography, music and Amateur Radio.

INSTRUCTOR'S MANUAL — $3.00
Necessary for instruction of Bugbook I and II. Answers questions regarding experiments, suggestions for further reading, philosophy of authors approach to digital electronics. A must for self-teaching individuals.

OP AMP MANUAL by Howard M. Berlin W3HB — $9.00
An experiment guide to application of operational amplifiers. Over 25 experiments on all phases of Op Amps.

DBUG — $5.00
8080 interpretive debugger. A program for entering, debugging and storing assembly language programs.

COMPLETE MANUAL FOR DIGITAL CLOCKS by John Weiss and John Brooks
Familiarizes technician or hobbyist with basic theories behind digital clocks. Includes trouble shooting guides, basic characteristics of clocks, soldering techniques, clock-component data sheets and construction tips. — $3.95

DISCRETE LEDS

.125" dia.

XC209	Red	5/$1
XC209	Green	4/$1
XC209	Orange	4/$1
XC209	Yellow	4/$1

.200" dia.

XC22	Red	5/$1
XC22	Green	4/$1
XC22	Yellow	4/$1
XC22	Orange	4/$1
SSL-22	RT	4/$1

.185" dia.

XC526	Red	5/$1
XC526	Red	100/$8
XC526	Green	4/$1
XC526	Yellow	4/$1
XC526	Clear	4/$1

.200" dia.

XC556	Red	5/$1
XC556	Red	100/$8
XC556	Green	4/$1
XC556	Yellow	4/$1
XC556	Orange	4/$1
XC556	Clear	7/$1

.190" dia.

XC111	Red	5/$1
XC111	Green	4/$1
XC111	Yellow	4/$1
XC111	Orange	4/$1

.085" dia. MV50 - Red - 6/$1
.170" dia. MV10 Red - 4/$1
INFRA-RED LED ¼" x ¼" x 1/16 Flat 5/$1.00

DISPLAY LEDS

TYPE	POLARITY	HT	PRICE	TYPE	POLARITY	HT	PRICE
MAN 1	Common Anode-red	.270	2.95	MAN 6680	Common Cathode-orange	.560	.99
MAN 2	5 x 7 Dot Matrix-red	.300	4.95	MAN 6710	Common Anode-red-D.D.	.560	.99
MAN 3	Common Cathode-red	.125	.25	MAN 6730	Common Anode-red ±1	.560	.99
MAN 4	Common Cathode-red	.187	1.95	MAN 6740	Common Cathode-red-D.D.	.560	.99
MAN 52	Common Anode-green	.300	1.25	MAN 6750	Common Cathode-red ±1	.560	.99
MAN 71	Common Anode-red	.300	1.25	MAN 6760	Common Anode-red	.560	.99
MAN 72	Common Anode-red	.300	.99	MAN 6780	Common Cathode-red	.560	.99
MAN 74	Common Cathode-red	.300	1.25	DL701	Common Anode-red ±1	.300	.99
MAN 81	Common Anode-yellow	.300	.99	DL702	Common Cathode-red	.300	1.25
MAN 82	Common Anode-yellow	.300	.99	DL704	Common Cathode-red	.300	.99
MAN 84	Common Cathode-yellow	.300	.99	DL707	Common Anode-red	.300	.99
MAN 3620	Common Anode-orange	.300	.99	DL741	Common Anode-red	.600	1.25
MAN 3630	Common Anode-orange ±1	.300	.99	DL746	Common Anode-red ±1	.630	1.49
MAN 3640	Common Cathode-orange	.300	.99	DL747	Common Anode-red	.600	1.49
MAN 4610	Common Anode-orange	.300	.99	DL749	Common Cathode-red ±1	.630	1.49
MAN 4640	Common Cathode-orange	.400	.99	DL750	Common Cathode-red	.600	1.49
MAN 4710	Common Anode-red ±1	.400	.99	DL338	Common Cathode-red	.110	.35
MAN 4730	Common Anode-red	.400	.99	FND70	Common Cathode	.250	.69
MAN 4740	Common Cathode-red	.400	.99	FND359	Common Anode	.350	.75
MAN 4810	Common Anode-yellow	.400	.99	FND503	Common Cathode (FND500)	.500	.99
MAN 6610	Common Anode-orange-D.D.	.560	.99	FND507	Common Anode (FND510)	.500	.99
MAN 6630	Common Anode-orange	.560	.99	5082-7300	4 x 7 Sgl. Digit-RHDP	.600	19.95
MAN 6640	Common Cathode-orange-D.D.	.560	.99	5082-7302	4 x 7 Sgl. Digit-LHDP	.600	19.95
MAN 6650	Common Cathode-orange ±1	.560	.99	5082-7304	Overrange character (±1)	.600	15.00
MAN 6660	Common Anode-orange	.560	.99	5082-7340	4 x 7 Sgl. Digit-Hexadecimal	.600	22.50

RCA LINEAR

CA3013	2.15	CA3082	2.00
CA2023	2.56	CA3083	1.60
CA3035	2.48	CA3086	.85
CA3039	1.35	CA3089	3.75
CA3046	1.30	CA3130	1.39
CA3059	3.25	CA3140	1.25
CA3060	3.25	CA3160	1.25
CA3080	.85	CA3401	.49
CA3081	2.00	CA3600	3.50

CALCULATOR CHIPS AND DRIVER

FCM3817	$5.00
MM5725	2.95
MM5736	1.95
MM5738	2.95
DM8864	2.00
DM8865	1.00
DM8887	.75
DM8889	.75
5030	7.95

CLOCK CHIPS

MM5309	$9.95
MM5311	4.95
MM5312	4.95
MM5314	4.95
MM5316	6.95
MM5318	9.95
MM5369	2.95
MM5841	9.95
7001	5.95

9374 - 7-segment LED driver common anode LEDs $ .99

IC SOLDERTAIL — LOW PROFILE (TIN) SOCKETS

	1-24	25-49	50-100		1-24	25-49	50-100
8 pin LP	$.17	.16	.15	22 pin LP	$ .37	.36	.35
14 pin LP	.20	.19	.18	24 pin LP	.38	.37	.36
16 pin LP	.22	.21	.20	28 pin LP	.45	.44	.43
18 pin LP	.29	.28	.27	36 pin LP	.60	.59	.58
20 pin LP	.34	.32	.30	40 pin LP	.63	.62	.61

SOLDERTAIL STANDARD (TIN)

14 pin ST	$.27	.25	.24	28 pin ST	$ .99	.90	.81
16 pin ST	.30	.27	.25	36 pin ST	1.39	1.26	1.15
18 pin ST	.35	.32	.30	40 pin ST	1.59	1.45	1.30
24 pin ST	.49	.45	.42				

SOLDERTAIL STANDARD (GOLD)

8 pin SG	$.30	.27	.24	24 pin SG	$ .70	.63	.57
14 pin SG	.35	.32	.29	28 pin SG	1.10	1.00	.90
16 pin SG	.38	.35	.32	36 pin SG	1.75	1.40	1.26
18 pin SG	.52	.47	.43	40 pin SG	1.75	1.59	1.45

WIRE WRAP SOCKETS (GOLD) LEVEL #3

8 pin WW	$.40	.38	.35	22 pin WW	$ .95	.85	.75
10 pin WW	.45	.41	.37	24 pin WW	1.05	.95	.85
14 pin WW	.39	.38	.37	28 pin WW	1.40	1.25	1.10
16 pin WW	.43	.42	.41	36 pin WW	1.59	1.45	1.30
18 pin WW	.75	.68	.62	40 pin WW	1.75	1.55	1.40

50 PCS. RESISTOR ASSORTMENTS $1.75 PER ASST.

ASST. 1	5 ea.	10 OHM 27 OHM	12 OHM 33 OHM	15 OHM 39 OHM	18 OHM 47 OHM	22 OHM 56 OHM	1/4 WATT 5%	50 PCS.
ASST. 2	5 ea.	68 OHM 180 OHM	82 OHM 220 OHM	100 OHM 270 OHM	120 OHM 330 OHM	150 OHM 390 OHM	1/4 WATT 5%	50 PCS.
ASST. 3	5 ea.	470 OHM 1.2K	560 OHM 1.5K	680 OHM 1.8K	820 OHM 2.2K	1K 2.7K	1/4 WATT 5%	50 PCS.
ASST. 4	5 ea.	3.3K 8.2K	3.9K 10K	4.7K 12K	5.6K 15K	6.8K 18K	1/4 WATT 5%	50 PCS.
ASST. 5	5 ea.	22K 56K	27K 68K	33K 82K	39K 100K	47K 120K	1/4 WATT 5%	50 PCS.
ASST. 6	5 ea.	150K 390K	180K 470K	220K 560K	270K 680K	330K 820K	1/4 WATT 5%	50 PCS.
ASST. 7	5 ea.	1M 2.7M	1.2M 3.3M	1.5M 3.9M	1.8M 4.7M	2.2M 5.6M	1/4 WATT 5%	50 PCS.

ASST. 8R Includes Resistor Assortments 1-7 (350 PCS.) **$9.95 ea.**

$5.00 Minimum Order – U.S. Funds Only
California Residents – Add 6% Sales Tax
Spec Sheets – 25¢
1978A Catalog Available–Send 35¢ stamp

NEW NAME

Jameco ELECTRONICS

a division of JAMES ELECTRONICS of California

PHONE ORDERS WELCOME (415) 592-8097

MAIL ORDER ELECTRONICS – WORLDWIDE
1021 HOWARD AVENUE, SAN CARLOS, CA 94070
Advertised Prices Good Thru August

WIRE-WRAP KIT — WK-2-W

WRAP • STRIP • UNWRAP

- Tool for 30 AWG Wire
- Roll of 50 Ft. White or Blue 30 AWG Wire
- 50 pcs. each 1", 2", 3" & 4" lengths — pre-stripped wire.

$12.95

WIRE WRAP TOOL WSU-30

WRAP • STRIP • UNWRAP - $6.95

WIRE WRAP WIRE — 30 AWG

25 ft. min. $1.25 50 ft. $1.95 100 ft. $2.95 1000 ft. $15.00
SPECIFY COLOR — White - Yellow - Red - Green - Blue - Black

WIRE DISPENSER — WD-30

- 50 ft. roll 30 AWG KYNAR wire wrap wire **$3.95 ea.**
- Cuts wire to desired length
- Strips 1" of insulation **Specify** — Blue-Yellow-White-Red

REPLACEMENT DISPENSER SPOOLS FOR WD 30
Specify blue, yellow, white or red **$1.98/spool**

EXAR

XR2206KA $14.95
Function Generator Kit (includes chip, P.C. Board and instructions)

XR2206KB $19.95
Function Generator Kit (includes all components, P.C. Board and instructions)

XR-L555 $1.50
Micro-Power version of the popular 555 Timer and directly interchangeable. Dissipates 1/15th the power and operates down to 2.7 volts. Perfect for battery operation and CMOS circuits.

XR2242CP $1.50
Precision timing circuit for generating timing pulses in **minutes, hours** and days or up to 1 year by using two. Reduces cost of time delay circuits. Basic 555 Timer with built-in 8-bit Counter.

XR205	$8.40	XR1489	1.39	XR2556	$3.20
XR210	4.40	XR1800	3.20	XR2567	2.99
XR215	4.40	XR2206	4.40	XR3403	1.25
XR320	1.55	XR2207	3.85	XR4136	1.25
XR555	.39	XR2208	5.20	XR4151	2.85
XR556	.99	XR2209	1.75	XR4194	1.45
XR567CP	.99	XR2211	5.25	XR4202	3.60
XR567CT	1.25	XR2212	4.35	XR4212	2.05
XR1310P	1.30	XR2240	3.45	XR4558	.75
XR1468CN	3.85	XR2264	4.25	XR4739	1.15
XR1488	1.39			XR4741	1.47

ZENERS — DIODES — RECTIFIERS

TYPE	VOLTS	W	PRICE	TYPE	VOLTS	W	PRICE
1N746	3.3	400m	4/1.00	1N4005	600 PIV	1 AMP	10/1.00
1N751A	5.1	400m	4/1.00	1N4006	800 PIV	1 AMP	10/1.00
1N752	5.6	400m	4/1.00	1N4007	1000 PIV	1 AMP	10/1.00
1N753	6.2	400m	4/1.00	1N3600	50	200m	6/1.00
1N754	6.8	400m	4/1.00	1N4148	75	10m	15/1.00
1N959	8.2	400m	4/1.00	1N4154	35	10m	12/1.00
1N965B	15	400m	4/1.00	1N4305	75	25m	20/1.00
1N5232	5.6	500m	.28	1N4734	5.6	1w	.28
1N5234	6.2	500m	.28	1N4735	6.2	1w	.28
1N5235	6.8	500m	.28	1N4736	6.8	1w	.28
1N5236	7.5	500m	.28	1N4738	8.2	1w	.28
1N456	25	40m	6/1.00	1N4742	12	1w	.28
1N458	150	7m	6/1.00	1N4744	15	1w	.28
1N485A	180	10m	5/1.00	1N1183	50 PIV	35 AMP	1.60
1N4001	50 PIV	1 AMP	12/1.00	1N1184	100 PIV	35 AMP	1.70
1N4002	100 PIV	1 AMP	12/1.00	1N1185	150 PIV	35 AMP	1.70
1N4003	200 PIV	1 AMP	12/1.00	1N1186	200 PIV	35 AMP	1.80
1N4004	400 PIV	1 AMP	12/1.00	1N1188	400 PIV	35 AMP	3.00

SCR AND FW BRIDGE RECTIFIERS

C36D	15A @ 400V	SCR(2N1849)	$1.95
C38M	35A @ 600V	SCR	1.95
2N2328	1.6A @ 300V	SCR	.50
MDA 980-1	12A @ 50V	FW BRIDGE REC	1.95
MDA 980-3	12A @ 200V	FW BRIDGE REC	1.95

TRANSISTORS

C106B1	.50			2N3904	4/1.00
MPSA05	.30	2N3055	.89	2N3905	4/1.00
MPSA06	5/1.00	MJE3055	1.00	2N3906	4/1.00
TIS97	6/1.00	2N3392	5/1.00	2N4013	3/1.00
TIS98	6/1.00	2N3398	5/1.00	2N4123	6/1.00
TIS133	5/1.00	PN3567	3/1.00	PN4249	4/1.00
TIS135	5/1.00	PN3568	4/1.00	PN4250	4/1.00
40409	1.75	PN3569	4/1.00	2N4400	4/1.00
40410	1.75	MPS3638A	5/1.00	2N4401	4/1.00
40673	1.75	MPS3702	5/1.00	2N4402	4/1.00
2N918	4/1.00	2N3704	5/1.00	2N4403	4/1.00
2N2219A	3/1.00	MPS3704	5/1.00	2N4409	5/1.00
2N2221A	4/1.00	2N3705	5/1.00	2N5086	4/1.00
2N2222A	5/1.00	MPS3705	5/1.00	2N5087	4/1.00
2N2369	5/1.00	2N3706	5/1.00	2N5088	4/1.00
2N2369A	4/1.00	MPS3706	5/1.00	2N5089	4/1.00
MPS2369	5/1.00	2N3707	5/1.00	2N5129	5/1.00
2N2484	4/1.00	2N3711	5/1.00	PN5134	5/1.00
2N2906	4/1.00	2N3724A	.65	PN5138	5/1.00
2N2907	5/1.00	2N3725A	1.00	2N5139	5/1.00
2N2925	5/1.00	2N3772	2.25	2N5210	5/1.00
MJE2955	1.25	2N3823	1.00	2N5449	3/1.00
2N3053	2/1.00	2N3903	4/1.00	2N5951	3/1.00

CAPACITOR CORNER

50 VOLT CERAMIC DISC CAPACITORS

	1-9	10-99	100		1-9	10-99	100-
10 pf	.05	.04	.03	.001µF	.05	.04	.035
22 pf	.05	.04	.03	.0047µF	.05	.04	.035
47 pf	.05	.04	.03	.01µF	.05	.04	.035
100 pf	.05	.04	.03	.022µF	.06	.05	.04
220 pf	.05	.04	.03	.047µF	.06	.05	.04
470 pf	.05	.04	.035	.1µF	.12	.09	.075

100 VOLT MYLAR FILM CAPACITORS

.001mf	.12	.10	.07	.022mf	.13	.11	.08
.0022	.12	.10	.07	.047mf	.21	.17	.13
.0047mf	.12	.10	.07	.1mf	.27	.23	.17
.01mf	.12	.10	.07	.22mf	.33	.27	.22

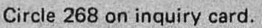
+20% DIPPED TANTALUMS (SOLID) CAPACITORS

.1/35V	.28	.23	.17	1.5/35V	.30	.26	.21
.15/35V	.28	.23	.17	2.2/25V	.31	.27	.22
.22/35V	.28	.23	.17	3.3/25V	.31	.27	.22
.33/35V	.28	.23	.17	4.7/25V	.32	.28	.23
.47/35V	.28	.23	.17	6.8/25V	.36	.31	.25
.68/35V	.28	.23	.17	10/25V	.40	.35	.29
1.0/35V	.28	.23	.17	15/25V	.63	.50	.40

MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS

Axial Lead				Radial Lead			
.47/50V	.15	.13	.10	.47/25V	.15	.13	.10
1.0/50V	.16	.14	.11	.47/50V	.16	.14	.11
3.3/50V	.14	.12	.09	1.0/16V	.15	.13	.10
4.7/25V	.15	.13	.10	1.0/25V	.16	.14	.11
10/25V	.15	.13	.10	1.0/50V	.16	.14	.11
10/50V	.16	.14	.12	4.7/16V	.15	.13	.10
22/25V	.17	.15	.12	4.7/25V	.15	.13	.10
22/50V	.24	.20	.18	4.7/50V	.16	.14	.11
47/25V	.19	.17	.15	10/16V	.14	.12	.09
47/50V	.25	.21	.19	10/25V	.15	.13	.10
100/25V	.24	.20	.18	10/50V	.16	.14	.12
100/50V	.35	.30	.28	47/50V	.24	.21	.19
220/25V	.32	.28	.25	100/16V	.19	.15	.14
220/50V	.45	.41	.38	100/25V	.24	.20	.18
470/25V	.33	.29	.27	100/50V	.35	.30	.28
1000/16V	.55	.50	.45	220/16V	.23	.17	.16
2200/16V	.70	.62	.55	470/25V	.31	.28	.26

Circle 200 on inquiry card.



Circle 268 on inquiry card.

Circle 133 on inquiry card.

Circle 11 on inquiry card.

Circle 193 on inquiry card.

Circle 395 on inquiry card.

Circle 62 on inquiry card.

Circle 171 on inquiry card.

Circle 312 on inquiry card.

Circle 308 on inquiry card.

RONDURE COMPANY

2522 BUTLER ST.
DALLAS, TEXAS 75235
214–630-4621

the computer room



ASCII Selectric with ASCII parallel electronics.
Immediate Delivery—Shipped from inventory.

ASCII SELECTRIC

SPECIAL SALE
$875.00

TESTED WITH NEW ASCII ELECTRONICS

Printer Mechanism: Heavy duty input/output, Series 745.

Weight: 120 lbs. Dimensions: 29"Hx35"Wx33"D.

Print Speed: (14.8 characters per second)

Platen: 15" wide, pin feed or form feed device optional (132 print positions).

Parallel output only—15 characters per second accepts 7 bit ASCII parallel w/strobe & prints on Selectric. The unit still works as a typewriter in off-line mode.

DATEL SELECTRIC (IBM Selectric Mechanism)



Specifications:
- Size: 21"Wx21"Dx8"H.
- Power Input 115 Volt Hz
- Inferface: RS232
- Weight: 54 lbs. (Shipping weight 65 lbs.)
- 15" Carriage
- 15 CPS
- Correspondence code
- Half Duplex
- 132 Print Positions, 10 Pitch

Used – $395

NOVATION DC3102A

Used
Working

$150.00

RS232 Connection
300 Baud

TI 990/4

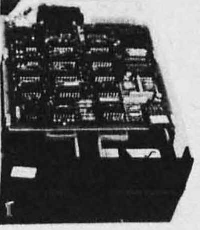
Single Board 16 Bit Micro Computer

NEW $250.00

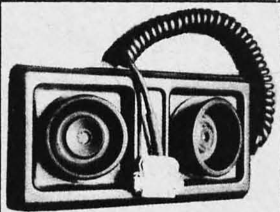
USED MODEMS & COUPLERS

NOVATION TC102 (Acoustic)	$45.00
AJ 233 (Acoustic)	$25.00
CF 318 (Hard Wire)	$25.00

SHUGART MINI-FLOPPY DRIVE



NEW PRICE
$325.00 ea.
Model SA-400



FLAT PACK ACOUSTICAL MODEM PICK-UP

Useable with most modem chips/kits
Used — $17.50 (w/prints)

ORDERING INFORMATION:
We ship the same day we receive a certified check or money order.
Texas residents add 5% sales tax. Please call if you have a question.
Write for our CATALOG of many parts, terminals, printers, etc.
All items subject to availability. Your money returned if we are out of stock.

SHIPPING INFORMATION:
Modems: $2.00 each; 2 for $4.00 UPS.
Large Items & Parts: Specify Freight or Air Freight Collect
Foreign Orders: Add appropriate freight or postage.
We now take Master Charge and Visa orders, Specify full number, bank number and expiration date.

F. Reichert Sales

1110 E. GARVEY AVE.
W. COVINA, CA 91790

- $20 MINIMUM ORDER
- FREE UPS DELIVERY ON U.S. ORDERS ONLY · OR BY SURFACE MAIL IF SPECIFIED
- MONEY BACK GUARANTEE (MOS & LED DEVICES EXCLUDED)
- UNDER 8 HOUR PROCESSING ON MONEY ORDERS & CASHIERS CHECKS.
- SORRY! NO COD'S, PO'S, OR CREDIT CARDS
- CALIF. RES. ADD 6% SALES TAX
- CANADA, PUERTO RICO & U.S. POSSESSIONS ADD U.S. $3.50. ALL OTHERS ADD U.S. $7.00
- U.S. FUNDS ONLY!

THE S-100 BUS DIAGNOSTIC BOARD

CENTURION IV

A LEGION OF DIAGNOSTIC TOOLS ON ONE BOARD!

- EXTENDER BOARD
- PULSE DETECTOR
- LOGIC PROBE
- CURRENT SHUNT
- CONVERTIBLE INTO BACKPLANE.
- ON BOARD CONNECTORS FOR SCOPE, DVM, VOM, GENERATORS, ETC.
- ON BOARD 5 WAY CONNECTORS AND RF CONNECTORS.
- OVER 160 WIRE-WRAP POINTS FOR TOTAL AND COMPLETE FLEXIBILITY IN WIRE WRAPPING ANY I/O CONNECTOR AND ALL LOGIC PROBE INPUTS INTO **ANY** AND **ALL** 100 BUS LINES, WHICH MEANS YOU CAN MONITOR AND MEASURE ANY PARAMETER ON ANY ONE OF THE 100 BUS LINES, OR LOGIC LEVEL PROBE **ANY FIFTY** BUS LINES SIMULTANEOUSLY. THE ONLY TOOL OF ITS KIND. A MUST FOR ANY SERIOUS SERVICE EFFORT.

Kit $119.00 Assembled $149.00

S-100 32K STATIC MEMORY BOARD

WITH THIS BOARD YOU CAN BEGIN WITH JUST 8K OF MEMORY AND ADD ON LATER. IT CAN BE AS SIMPLE AS INSERTING MORE MEMORY CHIPS.

FEATURES

1. FULLY STATIC — USABLE WITH ALL **DMA** DEVICES.
2. BUFFERED WITH NOISE SUPPRESSED INPUTS.
3. POPULATED IN 1K INCREMENTS (EACH 1K CAN BE ACCESSED INDIVIDUALLY).
4. INCLUDES A RELIABLE SINGLE SOURCE +5 VOLT REGULATOR.
5. PROM COMPATIBLE: THE BOARD MONITOR CAN BE EXCHANGED (PIN BY PIN WITHOUT BOARD MODIFICATION) WITH A BIPOLAR PROM PROGRAMMED TO YOUR SPECIFICATIONS.
6. CREATES NO WAIT STATES.
7. PRICES QUOTED ARE FOR 300nS 200mW MEMORIES.

— ASSEMBLED UNITS ARE FULLY TESTED AND BURNED IN.
— KITS AND BARE BOARD COME WITH FULL ASSEMBLY AND CHECKOUT INSTRUCTIONS.

	8K	16K	24K	32K	BARE BOARD
KIT	$270.00	$440.00	$580.00	$695.00	
ASSEMBLED	$296.00	$465.00	$612.00	$740.00	$38.00

MAXI SWITCH KEYBOARDS

UNENCODED-MOUNTED ON G-10 GLASS EPOXY BOARDS-A BLACK METAL FRAME KEEPS KEY SWITCHES SECURELY IN PLACE.

NO. 1

—53 KEY MAIN KEYBOARD
—10 AUXILLARY & CURSOR CONTROL KEYS
—11 KEY NUMERIC PAD
—BANK OF 5 AUXILLARY POWER & CONTROL, ROCKER ARM SWITCHES, ONE OF THEM LIGHTS UP.

$39.95

W.W. SOCKET CONNECTOR FOR NO. 1 KEYBOARD $2.95

NO. 2

—53 KEY KEYBOARD
—1 AUXILLARY POWER/ CONTROL DPDT ROCKER ARM SWITCH

$29.95

BEIGE METAL FRAME FOR NO. 2 $9.95

SOCKET CONNECTOR FOR NO. 2 KEYBOARD $2.95

ASCII KEYBOARDS

FULLY ASSEMBLED
FULLY FUNCTIONAL

—MAIN KEYBOARD SECTION OF 58 KEYS
—A HEX PAD OF 15 KEYS
—16 MORE PERIPHERAL KEYS
—ASCII ENCODED

NO. 1 - WITH ON BOARD UV PROM $99.95

NO. 2 - IDENTICAL TO THE ONE ABOVE, BUT WITHOUT ON BOARD UV PROM $79.95

WANT PEACE OF MIND? IN CRITICAL APPLICATIONS USE BIPOLARS!!

BIPOLAR RAMS

82S10	1024x1	$4.95
82S11	1024x1	$4.95

BIPOLAR PROGRAMMABLE LOGIC ARRAYS

82S100	$11.95
82S101	$11.95

BIPOLAR PROMS

8223	256 (32 x 8)	$2.65
82S23	256 (32 x 8)	$3.40
82S126	1024 (256 x 4)	$4.95
93427	256 x 4 TRISTATE	$4.95
82S131	2048 (512 x 4)	$6.95
6306-1	2K TRISTATE	$5.45
3604	4096 (512 x 8)	$11.95

ULTRA LOW POWER

256 x 4 · BIT STATIC CMOS RAM

5101L-3 650nS 30mW

DATA RETENTION CURRENT 0.7uA AT 2 VOLTS

$9.95

S-100 EXTENDER BOARD

WITH CONNECTOR

A MUST for trouble-shooting your Computer boards $17.95

S-100 EXPANDABLE MOTHER BOARD

- 8 SLOT EXPANDABLE BACKPLANE
- QUIET—GROUND PLANE DECOUPLES ALL SIGNAL LINES
- RELIABLE GOLD CONTACT CONNECTORS

COMPLETE KIT $66.00 ASSEMBLED $89.00

S-100 100 PIN

EDGE CARD CONNECTOR GOLD CONTACTS

$4.25

WIRE WRAP POST

2 LEVEL
10/78c
100/$6.50

4 LEVEL
10/98c
100/$7.40

HI-REL GOLD **WIRE WRAP SOCKET POST**

10/$1.20
100/$9.90
1000/$79.00

LIST PRICE 29c

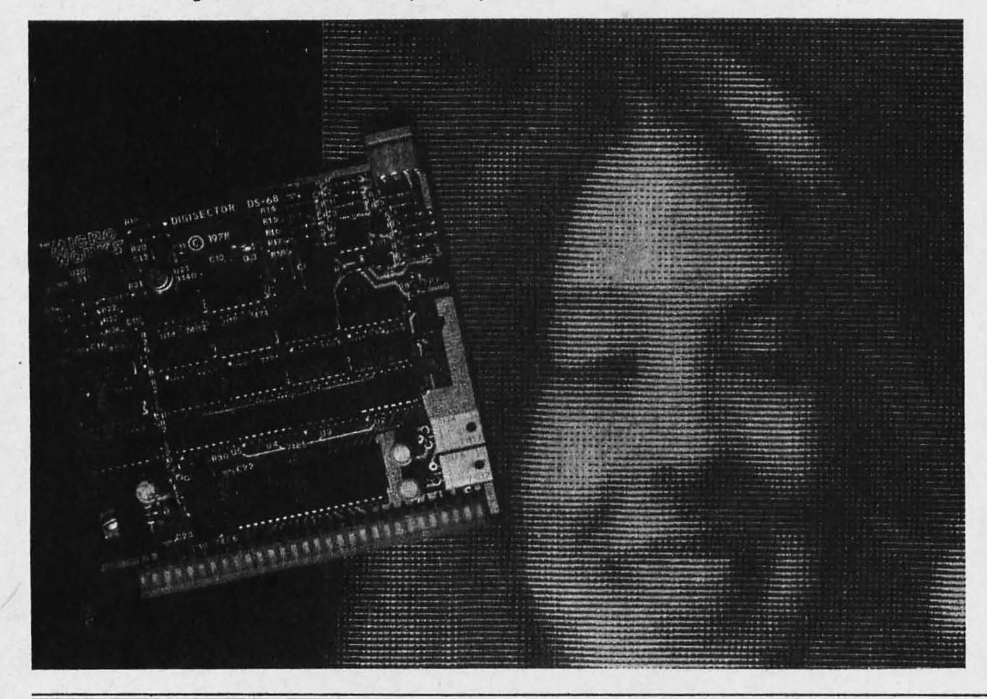
SUMMER SPECIALS! LOW PROFILE-SOLDERTAIL I.C. SOCKETS. OTHER SIZES IN PREVIOUS AD.

	GOLD INLAID	TIN
24 PIN	5/$1.09	5/$0.89
28 PIN	5/$1.19	5/$0.99
40 PIN	4/$1.19	4/$0.99

PRICES AND AVAILABILITY SUBJECT TO CHANGE WITHOUT NOTICE.

Circle 207 on inquiry card.

Give Sight to Your 6800 Computer System



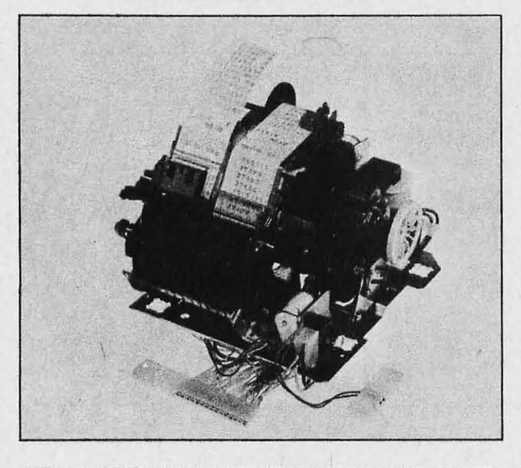
The Digisector (DS-68) functions in conjunction with an inexpensive television camera to present the computer with a high resolution digitized picture of the scene in view of the camera lens. The Digisector requires one IO slot in the SwTPC 6800 computer (or equivalent) and accepts either interlaced (NTSC) or noninterlaced (industrial) sync pulses from the video source. It features 256 by 256 picture element resolution, with up to 64 levels of grey scale. Data conversion times vary with resolution requirements but can be as low as 3 μs per picture element.

Applications include precision security systems, moving target indicators, computer portraiture, and fast to slow scan conversion for ham radio operators. With clever software, the Digisector can read paper tape, punched cards, strip charts, bar codes and musical scores.

The Digisector comes fully assembled, tested and burned in. The DS-68 is priced at $169.95. Software for computer portraiture and slow scan television is included. For further information, contact Micro Works, POB 1110, Del Mar CA 92014.■

Circle 615 on inquiry card.

Dual Printer from Addmaster



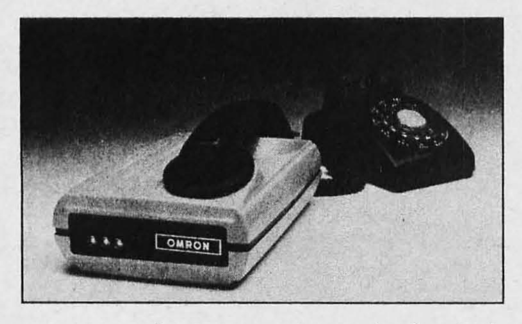
This dual printer prints three lines per second and has 11 character locations per column with a dual capacity of 6 to 10 columns, up to 22 columns as a single. A large library of characters is available.

Among the features are a small print mechanism, self-inking ribbon, variable paper space control. A tape rewind (one or both sides) is available.

The Addmaster dual printer operates with a 117 VAC 1.3 A drive motor. Other voltages are available. Quantity prices start at $66. For further information contact Addmaster Corp, 416 Junipero Serra Dr, San Gabriel CA 91776.■

Circle 616 on inquiry card.

Acoustic Coupler Aids Clarity



A new acoustic coupler (modem), designed to enhance clarity and accuracy of telephone line data transmission to and from computer terminals, has been introduced by Information Products Division of Omron Electronics Inc, 432 Toyama Dr, Sunnyvale CA 94086. The series 8300 Data Modems are designed for acoustic and hardware operation via the switched telephone network or private line installation.

Acoustically coupled with a Western Electric 500 handset or equivalent, the 8300 modems offer a convenient method of data transmission.

The 20 mA current loop interface (plug compatible with a DECwriter LA36) and a separate 25 pin RS-232C EIA connector may be operated simultaneously. The series is available as originate-only or originate-and-answer switch selectable for 300 bps operation. The modem is priced at $275 for single units.■

Circle 617 on inquiry card.

IO Processing Unit

A general purpose IO processing board, the APU100, which provides a high performance to the standard S-100 bus has been announced by Extensys Corporation. Designated the Extensys Asynchronous Processing Unit, the APU100 includes an on board 8080 processor. The unit operates asynchronously with the central processing unit of the computer system and transfers information by use of direct memory access.

The APU100 uses the system clock on the bus to provide internal timing so that all system processors are synchronized. The unit has 8192 bytes of programmable memory storage operating at 300 ns access time and 1024 bytes of 2708 type erasable read only memory storage in addition to its dedicated 8080 processor.

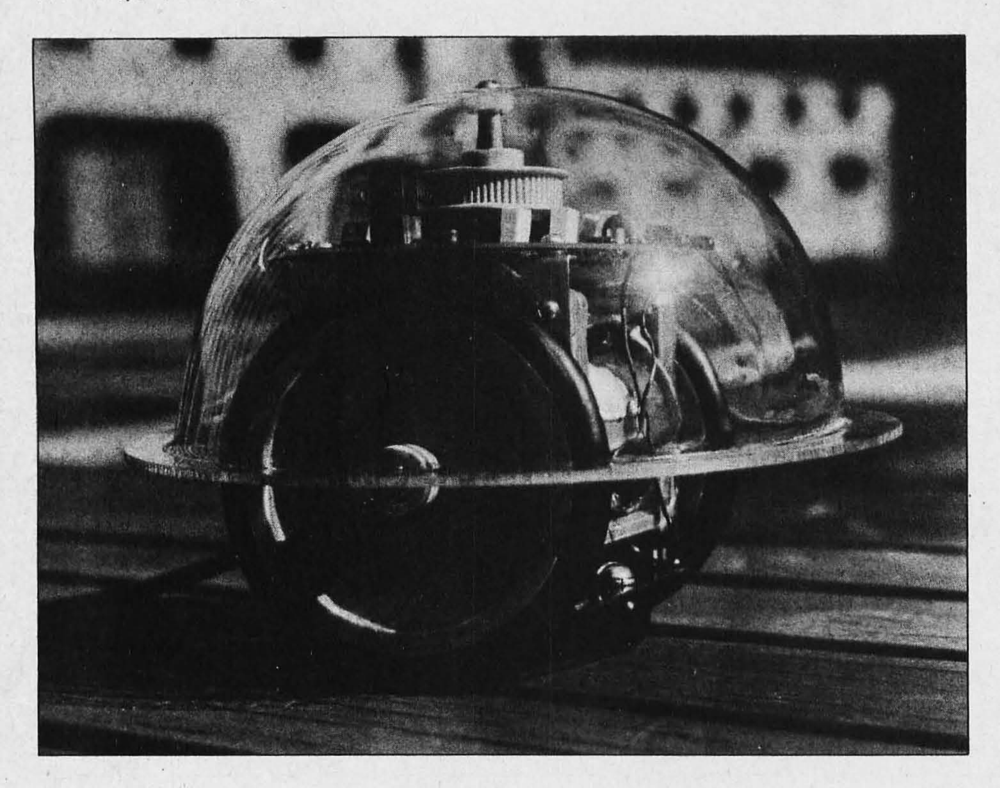
Using the APU100 frees up 8 K bytes of system memory by moving IO routines to the APU100, allowing more memory for application programs. System performance is improved with the opportunity to buffer up information using direct access memory. Slow speed IO devices can be serviced at their rated speeds while system operation continues at normal speed.

When used with Extensys' MM-16 Memory Manager, the APU100 performs in a high speed direct access memory mode, transferring greater chunks of data at full memory speeds.

Contact Extensys Corp, 380 Bernardo Av, Mountain View CA 94040, for additional information. ■

Circle 618 on inquiry card.

The Terrapin Inc Turtle...



Here are some more details about the new Terrapin Turtle, manufactured by Terrapin Inc, 33 Edinborough St, 6th Floor, Boston MA 02111. The information is condensed into the form of a photograph of the new peripheral which accompanies this first formal press release from the new company. Without bending the truth (too much), it could be claimed that this peripheral (excluding the computer required to drive it through a cable) is the world's least expensive commercially sold mobile general purpose robot if we exclude certain dedicated office or hospital delivery robots (at tens of thousands of dollars) and prototype R2D2-like forms seen in a pen (see page 16 of July 1978 BYTE) at the West Coast Computer Faire but never seen on the market to date in the advertising or literature which passes this desk. The Terrapin Turtle, a small electronic robot controllable by microprocessor, can "walk" (roll), touch (with its 3 1/2 inch radius hemispherical dome), and draw (lowering its pen attachment), as programmed. It has lights for eyes and a speaker to emit sounds. The Turtle requires a parallel interface: one compatible with an S-100 bus is available as an accessory. Each Turtle comes with 10 feet of cable and may be purchased either as a kit or fully assembled. Each kit comes with a tested, 20 page instruction manual.

The Turtle may be used to map rooms, solve mazes, teach simple geometry or programming concepts, as well as many other tasks. The Turtle is 5 inches high, crawls at 6 feet per second and is extremely versatile due to its touch sensors. Brochures are available. The kit costs $300; the assembled Turtle is $500; and the interface costs $40.■

Circle 529 on inquiry card.

Single Board Microperipherals from Burr-Brown



The MP810 and MP810-NS are two new microperipheral boards from Burr-Brown designed to accept up to 24 digital inputs. The boards are electrically and mechanically compatible with the Intel SBC-80, Intellec MDS and National BLC-80 microcomputers and operate from their +5 VDC supplies. They are programmed as memory locations and, with each input using one memory bit, any read command may be employed. When the board is read, logic 0 represents an open contact; logic 1, a closed contact. Each read command inputs the status of eight channels.

The MP810, with an on board power supply, operates with dry relay contacts and the MP810-NS, with voltage inputs, operates with wet relay contacts. Each group of eight inputs is isolated from other input groups and from computer bus to 600 VDC. Isolation between inputs of the MP810-NS is 300 VDC.

Input impedance is 15 k ohms and input delay is 25 μs maximum open to closed; 100 μs maximum closed to open. Minimum voltage needed to detect logic 1 is 17 V; logic 0, 4 V. For contact closure sense, maximum closed impedance is 6 k ohms; the minimum open impedance is 80 k ohms. Maximum voltage that can be applied across the MP810 input is 120 VAC RMS, or 60 VDC; across the MP810-NS inputs, 168 VAC RMS, or 84 VDC. In 1 to 9 quantities the MP810 is priced at $355, the MP810-NS at $295. For further information contact Burr-Brown, International Airport Industrial Park, Tucson AZ 85734.■

Circle 530 on inquiry card.

Intelligent Low Cost Terminal



The ZMS-50 is an intelligent terminal geared for a wide variety of data entry and text editing applications. It is a microcomputer based keyboard and video display unit controlled by Zentec provided programs executing out of read only memory. The terminal consists of a general purpose keyboard, 25 line by 80 character video display, 4 K or optional 16 K bytes of system programmable memory, and an asynchronous or synchronous RS-232C interface supported by a telecommunications firmware package. An optional simplex (output only) RS-232C interface to a local printer is also available. Priced under $2000 from Zentec Corp, 2400 Walsh Av, Santa Clara CA 95050.■

Circle 531 on inquiry card.

FREE PROBES FREE PROBES FREE PROBES

NLS MS-215 DUAL TRACE MINISCOPE $395.00



NLS

1¢ SALE

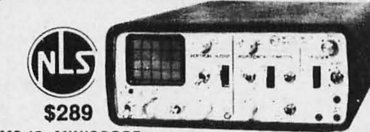
LM3A 3 dig 1% DC .... $125.
LM3.5A 3½ dig .5% DC .... $147.
LM40A 4 dig .1% DC.......... $190.
LM4A 4 dig .03% DC .... $227.

- Rechargeable batteries and charger included
- Measures DC Volts, AC Volts, Ohms and Current
- Automatic polarity, decimal and overload indication
- Rechargeable batteries and charger
- Measures DC Volts, AC Volts, Ohms and Current
- Automatic polarity, decimal and overload indication
- No zero adjustment and no full-scale ohms adjust
- Battery operated — NiCad batteries; also AC line operation.
- Large LED display for easy reading without interpolation
- Size: 1.9"H x 2.7"W x 4"D
  Parts & labor guaranteed 1 year
  Tilt stand option .... $ 3.50
- Leather case $16.00

Purchase any of the LM series Meters and buy the LEATHER CASE for 1¢

LAST CHANCE: NLS PRICES GOING UP

NLS



$289

MS-15 MINISCOPE *With Rechargeable Batteries & Charger Unit*

- 15 megahertz bandwidth.
- External and internal trigger.
- Time base — .1 microsec to 0.5 sec/div - 21 settings. ± 3%.
- Battery or line operation.
- Automatic & line sync modes.
  Power consumption 15 watts.
- Vertical gain — 10 mV to 50 V/div - 12 settings ± 3%.
- Viewing area 1" x 1.35".
- Size 2.7"H x 6.4"W x 7.5"D, 3 pounds.
- Parts & labor guaranteed 1 year
- 10:1 10 meg probe $24.50
  Leather carrying case $30.00
- MS-215 Dual Trace Version of MS-15 $395.00

PROBE 1¢

PROBE 1¢ with the purchase of SCOPE and the MENTION of this MAGAZINE

NLS

3 LEVEL GOLD WIRE WRAP SOCKETS

	1-24	25-49	50-99	100-249	250-999	1K-5K
8 pin*	.41	.38	.35	.31	[illegible]	.23
14 pin*	.39	.38	.36	.32	[illegible]	.27
16 pin*	[illegible]	.42	.39	.35	[illegible]	.30
18 pin	.63	.58	.54	.47	.42	.38
20 pin	.80	.75	.70	.68	.58	.53
22 pin	.90	.85	.80	.70	.61	.57
24 pin	.90	.84	.78	.68	.63	.58
28 pin	1.10	1.00	.90	.84	.76	.71
40 pin	1.50	1.40	[illegible]	1.20	1.04	.89

Sockets purchased in multiples of 50 per type may be combined for best price.

All sockets are GOLD 3 level closed entry. *End and side stacable. 2 level, Solder Tail, Low Profile, Tin Sockets and Dip Plugs available. CALL FOR QUOTATION

3 LEVEL GOLD WIRE WRAP SOCKETS

S-100 BUS EDGE CONNECTORS

SALE SALE

S100-WWG 50/100 Cont. .125 ctrs. 3 LEVEL WIRE WRAP .025" sq. posts on .250 spaced rows. GOLD plated.

1-4	5-9	10-24
$4.00	$3.75	$3.50

S100-WWN 50/100 Cont. .125 ctrs. 3 LEVEL WIRE WRAP .025" sq. posts on .250 spaced rows. NASGLO tin-nickel plated.

1-4	5-9	10-24
$3.50	$3.25	$3.00

S100-STG 50/100 Cont. .125 ctrs. DIP SOLDER TAIL on .250 spaced rows for VECTOR and IMASI motherboards GOLD plated.

1-4	5-9	10-24
$4.00	$3.75	$3.50

S100-STN 50/100 Cont. .125 ctrs. DIP SOLDER TAIL on .250 spaced rows for VECTOR and IMASI motherboards. NASGLO tin-nickel plated.

1-4	5-9	10-24
$3.50	$3.25	$3.00

RG81G 50/100 Cont. .125 ctrs. DIP SOLDER TAIL on .140 spaced rows for ALTAIR motherboards. GOLD plated. $5.00

R681-3 50/100 Cont. .125 ctrs. PIERCED SOLDER EYELET tails. GOLD $7.35

Other Popular Edge Connectors

R644-G 22/44 Cont. .156 ctrs. PIERCED SOLDER EYELES tails. GOLD plated.

1-4	5-9	10-24
$3.00	$2.75	$2.50

R644-3 22/44 Cont. .156 ctrs WIRE WRAP tails. GOLD $4.71

ATTN: OEM'S and Dealers, many other connectors available call or quotation.

8803 MOTHER BOARD FOR S100 BUS MICRO-COMPUTERS

Vector

- Kit includes 12 tantalum capacitors for +5, +12, —12 buses and insulated mounting spacers
- Wiring side shown. Component side bare epoxy glass with white markings for component locations
- G10 epoxy glass board with 2 ounce copper, solder plated and .038 diameter holes for leads
- Solder mask with solder windows on etched circuits to avoid accidental short circuits
- Mounts 11 receptacles with 100 contacts (2 rows) on .125 centers with .250 row spacing. Vector part number R681-2, or mounts 10 receptacles plus interconnections to smaller mother board for expansion
- Includes etched circuits and instructions for option of active, pull-up, or floating terminations
- Large buses +5V and GND (10 AMPS), ± 12V or 16V (7 AMPS). Current ratings are per MIL-STD-275 with 10°C rise
- Fits in Vector-pak enclosures
- Fits in IMSAI 8080 microcomputer as expander board

Price: $29.50



Vector **Plugboards**

8800V

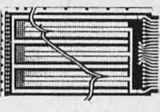
Universal Microcomputer/processor plugboard, use with S-100 bus. Complete with heat sink & hardware 5.3" x 10" x 1/16

1-4	5-9	10-24
$19.95	$17.95	$15.96

8801-1

Same as 8800V except plain, less power buses & heat sink

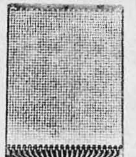
1-4	5-9	10-24
$14.95	$13.46	$11.96



3682 9.6" x 4.5" $10.97

3682-2 6.5" x 4.5" $9.81

Hi-Density Dual-In-Line Plugboard for Wire Wrap with Power & Grd. Bus Epoxy Glass 1/16" 44 pin con. spaced .156



3677 9.6" x 4.5"

$10.90

3677-2 6.5" x 4.5"

$9.74

Gen. Purpose D.I.P. Boards with Bus Pattern for Solder or Wire Wrap. Epoxy Glass 1/16" 44 pin con. spaced .156

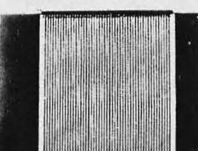
3662 6.5" x 4.5"

$7.65

3662-2 9.6" x 4.5"

$11.45

P pattern plugboards for IC's Epoxy Glass 1/16" 44 pin con. spaced .156



3690-12

CARD EXTENDER

Card Extender has 100 contacts-50 per side on .125 centers-Attached connector-is compatible with S-100 Bus Systems............$25.00

3690 6.5" 22/44 pin .158 ctrs. Extenders......$12.00

1/16 *Vector* BOARD

.042 dia holes on 0.1 spacing for IC's

Phenolic PART NO.	SIZE	PRICE 1-9	10-19
64P44XXXP	4.5x6.5"	$1.49	1.34
169P44XXXP	4.5x17"	$3.51	3.16
Epoxy Glass			
64P44	4.5x6.5"	$1.70	1.53
84P44	4.5x8.5"	$2.10	1.89
169P44	4.5x17"	$4.30	3.87
169P84	8.5x17"	$7.65	6.89

SLIT-N-WRAP

Wraps insulated wire on .025" square posts
FOUR TIMES FASTER
than regular manual wrap-post tools



P180 with two 100' spools of 28 ga. wire

$24.50

P160-4T includes charger, wire **$75.00**



Vector

NO PRE-STRIPPING*
NO PRE-CUTTING*
SPOOL-FED WIRE*

*The spooled wire passes through the tool past a slitting edge next to the wrap post. A narrow longitudinal cut is made in the insulation where it presses the square post corner, the bared copper is indented by the sharp edge (7 turns = 28 contacts). Insulation is slit where wrapped but not between terminal posts when fed straight out of the tool.

"Daisy-chain" runs. Reliable and easy to use. Manual or power operation

SLIT-N-WRAP WIRE $2.75 Package
NO. 28 GAGE INSULATED WIRE, 100' SPOOLS
W28-2-A Pkg. 3 Green W28-2-C Pkg. 3 Clear
W28-2-B Pkg. 3 Red W28-2-D Pkg. 3 Blue

2708
8K 450 ns
EPROM
FACTORY PRIME
$10.00 EA.
25 + *Call For Price*



WOW

14 & 16 PIN GOLD 3 LEVEL WIRE WRAP SOCKETS

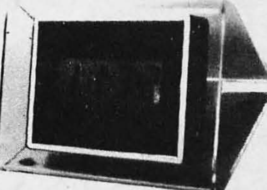
14 - G3 100 for **$30.00**

16-G3 100 for **$30.00**

50 of each for **$32.00**

Sockets are End & Side stackable, closed entry

LIQUID CRYSTAL DIGITAL CLOCK-CALENDAR



- For Auto, Home, Office
- Small in size (2x2½x½)
- Push button for seconds release for date.
- Clocks mount anywhere with either 3M double-sided tape or VELCRO, included.
- 2 MODELS AVAILABLE:
  LCD-101, portable model runs on self-contained batteries for better than a year.
  LCD-102, runs on 12 Volt system and is back-lighted.
- LCD-101 or LCD-102 your choice **$34.95 ea.**
- Clear desk stand for **$2.00**



LEDU MG 10A List $72.

Perfectly balanced, fluorescent lighting with precision magnifier lens. For prof'l, techn'l & hobbist. Has die cast protective shade, inst. start 3 diopter lens 42" reach. $44.95 with T-9 fluorescent lamp (required) $49.95. colors Gray or Black, Chocolate Brown. 2 choices please

$44.95 SPECIAL

FM-7 With Rechargeable Batteries & Charger Unit $195

SC-5 With Rechargeable Batteries & Charger Unit $89

NLS

Features include: • By using the new NLS SC-5 Prescaler, the range of the FM-7 Frequency Meter, which is 10 Hz to 60 MHz, may be extended to 512 MHz (the upper VHF & UHF frequency bands). • The FM-7 utilizes an LED readout, providing 7-digit resolution. • The FM-7 can be calibrated to an accuracy of 0.00001%. • The SC-5 is accurate to one part per million. • Each unit has 30 millivolts sensitivity, is battery powered and has a charger unit included. • Dimensions of each are 1.9" H x 2.7" W x 3.9" D. • The units may be obtained separately or as a "Frequency Duo." • Parts & Labor guaranteed 1 year.
Tilt stand option .... $ 3.50
Leather case $16.00

MICRO-KLIP
for .042 dia. holes
(all boards on this page)
T42-1 pkg. 100 $ 1.50
T42-1 pkg. 1000 ..... $11.00
P-149 hand installing tool ...... $ 2.03



.8" LED ALARM CLOCK

12 hr. LED Alarm Clock uses 3½ digit .8" LED Display with AM/PM indicators and colons. Direct drive. PIN to PIN interface with S1998A I.C. Just add switches. AC Supply. Alarm. Display and I.C. only.

$7.95 or 2/$15.00



Price Breakthrough! **$17.50**



MA1003 CAR CLOCK

Bright Green Fluorescent Display Crystal Time Base Assembled, just add switches and 12 VDC.

SPECIAL

14CS2 100 for $14.00
16CS2 100 for $16.00
14 pin CS2 10 for $2.00
16 pin CS2 8 for $2.00

These low cost DIP sockets will accept both standard width plugs and chips. For use with chips, the sockets offer a low profile height of only .125" above the board. These sockets are end stackable.

Vector

WRAP POST
for .042 dia. holes
(all boards on this page)
T-44 pkg. 100 .. $ 2.28
T-44 pkg. 1000....... $14.00
A-13 hand installing tool ...... $ 2.80

PRIORITY ONE ELECTRONICS ©

VISA

4911B West Rosecrans, Hawthorne, CA 90250

master charge

Terms: VISA, MC, BAC, check, Money Order, C.O.D., U.S. Funds Only. CA residents add 6% sales tax. Minimum order $10.00. Orders less than $75.00 include 10% shipping and handling, excess refunded. Just in case ... please include your phone no. **"Sorry, no over the counter sales"**

Prices subject to change without notice.

Send for our latest brochure. **phone orders welcome (213) 973-4876** OEM and Institutional inquiries invited.

24 PIN DIP PLUGS WITH COVERS

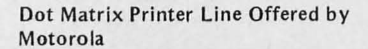


3 / $1.00
40 / $10.00

NLS MS-215 DUAL TRACE MINISCOPE $395.00

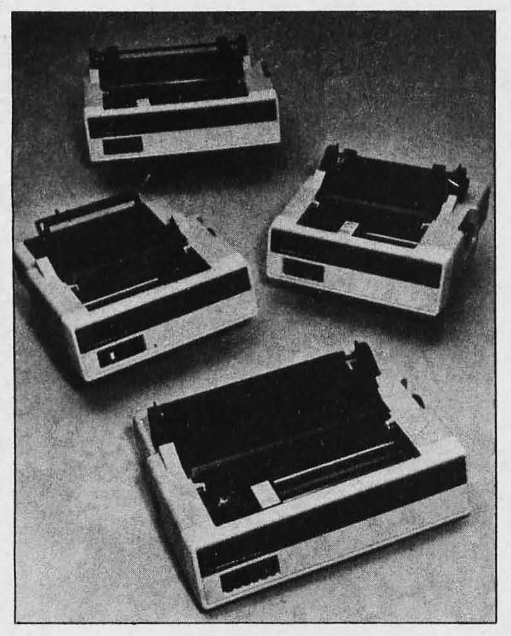
FREE PROBES FREE PROBES FREE PROBES

Circle 304 on inquiry card.



PERIPHERALS

Dot Matrix Printer Line Offered by Motorola

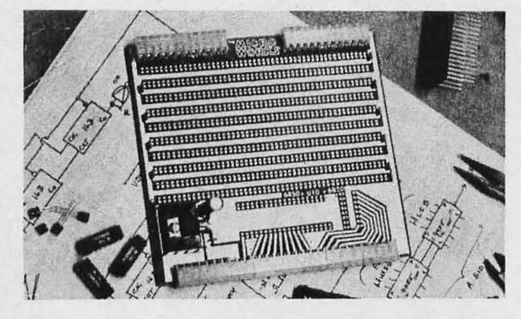


Motorola Microsystems, POB 20812, Phoenix AZ 85036, has announced a new line of four dot matrix printers. The line printers offer a full range of features, including 80 and 132 column formats; 60, 120 and 180 characters per second (cps); bidirectional and logic seeking print heads. All four printers are equipped with an interface IO module and an interconnection cable assembly that adapt them to the various Motorola microcomputer development systems, including the EXORciser and the EXORterm 100 and 200. These interface accessories permit the printers to be used with the company's line of Micromodules (microcomputer board systems and subsystems). Model 779 is a printer capable of printing from 80 to 132 columns of 5 by 7 dot matrix at rates of from 21 to 90 lines per minute at 60 cps.

The Model 781 is an 80 column character printer which features bidirectional logic seeking movement of the print head enabling throughput of up to 120 lines per minute. The Model 702 is also equipped with a bidirectional logic seeking print head with a head speed of 120 cps. This model has 132 character print columns and is capable of throughput of from 45 to 185 lines per minute (lpm). The Model 703 features a head speed of 180 cps and provides throughput rates of from 70 to 280 lpm. All models except the 779 have tractor feed with a paper out sensor and use standard computer paper, from one to six parts. Model 779 has pinch roll feed and uses standard Teletype roll paper. Prices are $1495 for the Model 779; $2095 for Model 781; $2500 for Model 702; and $3125 for Model 703.■

Circle 538 on inquiry card.

Universal IO Board



This IO board, called a universal IO board by the company, has space for a 40 pin wire wrap socket into which you can plug any of Motorola's 40 or 24 pin interface integrated circuits; the data and control lines are connected to the appropriate edge connector pins. All other bus connections are brought out to a 16 pin socket pad. A +5 V regulator and all Molex connectors are provided; regulated +5 V and ground are bussed among the locations for up to 35 14 pin integrated circuits. The price is $24.95 completely assembled and tested. Contact The Micro Works, POB 1110, Del Mar CA 92014.■

Circle 542 on inquiry card.

Multifont Printing Capability Added to Diablo Matrix Printers

The Model 24610-03 library card, in conjunction with the Diablo Model 2300 matrix printer, permits users to mix up to nine type fonts while printing a single document.

The card, which inserts into the printed circuit card cage of the printer, can be programmed by Diablo to add up to four different type and special symbol fonts to the standard font with each printer. In addition, the Model 2300 offers an optional plug-in read only memory which contains four additional foreign language or special fonts.

Each of the eight additional fonts available through implementation of the library card and the read only memory recognizes the full 128 character ASCII set and can print up to 96 of those characters.

Any combination of the eight additional fonts and the standard Diablo matrix font can be selected by the user under program control and can be mixed while printing. Among the standard fonts from which the user can select are German, Norsk, Scandia, Hebrew, French, French Canadian, and APL languages and symbols, in addition to high resolution ASCII and APL.

The cost of the Model 24610-03 libraby card in OEM quantities starts at $120. For further information, contact Diablo Systems Inc, 24500 Industrial Blvd, Hayward CA 94545.■

Circle 540 on inquiry card.

Serial Minifloppy Buffered Terminal



This intelligent buffered data terminal, Model IDS 3901, is offered by Interdyne Company, 14761 Califa St, Van Nuys CA 91411. It uses a 5¼ inch industry standard diskette drive and is RS-232C compatible. Average access time is 0.6 seconds. It has a data buffer holding up to 128 characters and is capable of being edited, a block rewrite capability and it allows insertion of blocks or entire paragraphs into previously written text. An automatic high speed block search and verify are included as well as character pattern search under operator control or prerecorded instructions. The IDS 3901 is controlled by 30 ASCII commands and outputs 13 English messages.

Other features include storage of 143 K bytes per diskette, switch selectable asynchronous transmission rates from 110 to 19,200 bps, ASCII text as well as transparent binary modes and auto error check and retry.

All this is contained in a stand alone desktop unit for connection to printers, videos, modems, computers and other terminals. By adding the IDS 3901 to a keyboard and display, users have an intelligent system with store and forward capability for program preparation and loading, data entry and storage, text editing and off line printing.

The IDS 3901 is priced at $2050.■

Circle 539 on inquiry card.

Product Update

In the June 1978 BYTE (page 178) we published a What's New? item describing the PET 100, an S-100 bus adapter for the Commodore PET computer. The manufacturer, HUH Electronic Music Productions, has notified us that the name has been changed to S-100 MPA. The product has been upgraded so that it meets the proposed IEEE specifications for the S-100 bus and it now can be a stand alone processor. The company's new address is 1429 Maple St, San Mateo CA 94402.■

Circle 541 on inquiry card.

POLY PAKS

COMPUTER PERIPHERAL & PARTS WORKSHOP

MAIL ORDER

POLY PAKS INTRODUCES THE FIRST MAGAZINE RETAIL STORE FOR THE ELECTRONIC MAIL-ORDER HOBBYIST!





KEYBOARD & ENCODER KIT

Cat. No. 8S5001 Kit $65.00
Cat. No. 8S5002 Wired $69.95

Outputs standard 7 bit ASCII; interfaces with most data systems. Keyboard pre-assembled onto PC board, 2 key rollover. Electronic shift lock and carriage return. 4 modes: Normal, control, shift, shift/control. Additional functions can be assigned by user. +5, -12VDC, 200 ma. Negative or positive logic, jumper selectable. Exlcusive test feature... 7 LED's display the ASCII code. Complete kit, nothing else to buy! Size: 13" x 5 1/2 x 1 1/4". 3 lbs.

- Uses MOS Encoder ROM!
- 7 LED Test Feature!
- 64 key keyboard!
- Encodes 128 ASCII Characters
- Interfaces with ALTAIR, IMSAI, and more!

HANDY COMPUTER MULTITESTER

$8.88

- 1000 ohms per volt

1% precision, movements diode protected against burnout. Measures DC volts 0-15-150-1000; AC volts 0-15-150-1000; DC current 0-150ma; resistance X1000. Sensitivity 1000 ohms/volt AC-DC. Uses penlite cell, not included. Size 2 3/8 x 3 1/2 x 1 3/8". Wt. 5 ozs.
Cat. No. 8S 3921

MICROPROCESSORS! MEMORIES! SUPPORT!

Order by Cat. No. 8S3459 and type

Type	Description	Sale
Z-80A	8 Bit CPU	$34.95
MC6800L	8Bit CPU	19.95
8080A	8 Bit CPU, 2 usec	12.95
8008	CPU	7.95
1101	256 x 1 Static RAM	.69
1103	1K Dynamic RAM	1.29
2102-L1	1K x 1 Lo-power RAM	1.69
2111	256 x 4 Static RAM	5.95
2112	256 x 4 Static RAM	2.49
1702A	256 x 8 EPROM	3.95
2708	8K EPROM	12.95
MK4200P11	4K Dyn RAM, 350 nsec	3.95
MK4096	4K Dyn RAM	3.95
MK4116	16K Dyn RAM	29.95
MM5202	2K PROM	2.95
MM5203	2K EPROM	8.95
MM5260	1K Dyn RAM	.99
MM5262	2K x 1 Dyn RAM	.99
8212	8 Bit I/O Port	3.95
8216	Bi-Direct Bus Driver	3.95
8224	Clock Gen	4.95
8228	System Cont	8.95
8251	Communication Int	11.50
8255	Periph Interface	11.95

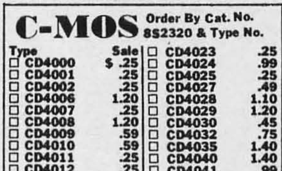
HEXADECIMAL MICROPROCESSOR AND CONTROL KEYBOARD KIT! 34.95

Address microprocessors, control computer operated equipment. 2 key rollover. Has 20 keys, 16 encoded, 4 external to be assigned by user. Output 4 bit binary. Also, an EXCLUSIVE FEATURE...4 leds display the binary output. TTL/CMOS compatible, requires +5, 12VDC. Complete kit! Nothing else to buy! With instructions.

Wt. 2 lbs.

- 4 bit binary/hexadecimal output plus strobe!
- New! Improved design! Saves you time!
- Keyboard pre-assembled onto PC board!

Cat. No. 8S5009 Hexadecimal Kit $34.95
Cat. No. 8S5010 Hexadecimal Wired $39.95



BRIDGE RECTIFIERS!

FULL WAVE

PIV	2 AMP (#8S1346)	6 AMP (#8S2456)	10 AMP (#8S2447)	25 AMP (#8S2273)
50	$ .59	$ .88	$1.05	$1.20
100	.65	.99	1.15	1.25
200	.69	1.19	1.29	1.95
400	.89	1.40	1.79	2.95
600	.99	1.69	1.95	3.95
800	1.19	1.95	2.25	4.95
1000	1.25	2.25	2.50	5.50

Orer by Cat. No. Amperage and Voltage

MIX-N-MATCH As low as $16.88 DUAL POWER SUPPLIES

- 1 Amp each voltage! 2 Amps total!

Cat. No.			Sale
8S3995	+5	-5	$16.88
8S3996	+5	-12	17.88
8S3999	+12	-12	18.88
8S4003	+15	-15	19.88

A new concept in power supply design. Choose from any one of four popular positive and negative combinations. Uses LM340 & LM320 series regulators and large heat sinks. Fits easily into gadget boxes or cabinet of your own design. Easy to assemble with stepbystep instructions. (Includes PC board, transformers, line cord, capacitors, rectifiers, heat sinks and h'ware). Complete kit, nothing else to buy. Wt. 10 lbs.

TTL'S

Type	Sale	Type	Sale
SN7400	$.14	SN7483	.55
SN7401	.14	SN7486	.29
SN7403	.14	SN7488	.79
SN7404	.14	SN7490	.45
SN7405	.14	SN7491	.59
SN7406	.14	SN7492	.39
SN7410	.14	SN7493	.39
SN7411	.19	SN7495	.49
SN7413	.49	SN7496	.29
SN7416	.25	SN74100	.89
SN7417	.24	SN74107	.29
SN7420	.19	SN74109	.65
SN7421	.25	SN74113	.39
SN7423	.25	SN74121	.31
SN7426	.19	SN74123	.49
SN7430	.14	SN74126	.49
SN7432	.24	SN74132	.71
SN7437	.24	SN74145	.69
SN7438	.24	SN74151	.69
SN7440	.19	SN74153	.69
SN7443	.49	SN74154	.99
SN7445	.69	SN74155	.79
SN7446	.65	SN74157	.59
SN7447	.65	SN74161	.79
SN7448	.65	SN74163	.79
SN7450	.14	SN74164	.79
SN7451	.14	SN74165	.99
SN7454	.19	SN74166	1.19
SN7460	.19	SN74173	.99
SN7462	.29	SN74174	.89
SN7470	.14	SN74175	.69
SN7472	.19	SN74177	.79
SN7473	.35	SN74179	1.99
SN7474	.35	SN74180	.49
SN7475	.39	SN74190	1.19
SN7476	.39	SN74192	.69
SN7480	.19	SN74193	.69
		SN74194	.79
		SN74197	.75

Low Power ICs

Order By Cat. No. 8S3667 & Type No.

Type	Sale	Type	Sale
74LS00	$ .29	74LS132	.99
74LS02	.29	74LS138	.99
74LS04	.29	74LS139	.99
74LS08	.29	74LS151	.89
74LS10	.29	74LS153	.99
74LS11	.36	74LS155	1.25
74LS13	.59	74LS160	.99
74LS14	.99	74LS161	.99
74LS20	.29	74LS162	.99
74LS21	.36	74LS163	.99
74LS22	.36	74LS168	1.25
74LS27	.39	74LS169	1.25
74LS30	.29	74LS173	1.25
74LS32	.35	74LS174	.99
74LS37	.39	74LS190	1.75
74LS38	.39	74LS191	1.75
74LS42	.79	74LS192	1.50
74LS47	.99	74LS193	1.25
74LS74	.49	74LS195	1.25
74LS90	.89	74LS196	1.25
74LS92	.89	74LS197	1.25
74LS93	.89	74LS257	.99
74LS109	.49	74LS258	.99
74LS112	.49	74LS266	.39
74LS113	.49	74LS366	.69
74LS114	.49	74LS368	.69
		74LS390	1.95

C-MOS

Order By Cat. No. 8S2320 & Type No.

Type	Sale	Type	Sale
CD4000	$ .25	CD4023	.25
CD4001	.25	CD4024	.99
CD4002	.25	CD4025	.25
CD4006	1.20	CD4027	.49
CD4007	.25	CD4028	1.10
CD4008	1.20	CD4029	1.20
CD4009	.59	CD4030	.45
CD4010	.59	CD4032	.75
CD4011	.25	CD4035	1.40
CD4012	.25	CD4040	1.40
CD4013	.45	CD4041	.99
CD4014	.99	CD4042	.89
CD4015	1.20	CD4044	.99
CD4016	.45	CD4046	1.50
CD4017	1.20	CD4047	2.50
CD4018	1.20	CD4049	.45
CD4019	.35	CD4053	.89
CD4020	1.25	CD4066	.90
CD4021	1.25	CD4071	.25
CD4022	1.25	CD4072	.39
		CD4077	.69

COMPUTER GRADE CAPACITORS

Order by Cat. No. 8S5112 and value!

Mfd	WVCD	Sale
8,000	50	$2.75
13,000	40	2.25
14,000	50	2.75
15,000	12	1.50
22,000	75	3.95
24,000	30	2.50
44,000	30	3.95

DISCRETE LED'S 6 for $1.19

Order in multiples of 6 of each type!

JUMBO .34" h
MEDIUM .24"
MICRO .21" (TO-18)

Cat. No.	Description	Similar to
8S2135	Jumbo Red	MV5053
8S1944	Jumbo Yellow	XC556Y
8S2138	Jumbo Green	XC556G
8S2785	Medium Red	XC22
8S2788	Medium Green	XC22G
8S2137	Micro Red	XC209
8S1948	Micro Yellow	XC209Y
8S2140	Micro Green	XC290G

SILICON POWER STUD RECTIFIERS!

Order by Cat. No. Amperage and voltage.

PIV	12 AMP (#8S727)	50 AMP (#8S727)	250 AMP (#8S685)
50	$ .29	$ .75	$4.95
100	.36	1.10	6.50
200	.45	1.35	7.50
400	.61	1.75	8.50
600	.79	2.25	10.50
800	.90	2.75	11.50

"RED" LED READOUTS!

as low as 20¢ each

Cat. No.	Size	Description	Sale
8S1890	.127"	MAN-3**	5 for $1.00
8S1503	.19	MAN-4*	2 for $1.19
8S1273	.27	MAN-1*	$1.00
8S3093	.33	MAN-72 equal*	$1.19
8S3161	.33	MAN-74 equal**	$1.19
8S3512	.35	FND359**	$1.00
8S2949	.5	FND500*	$1.50
8S2950	.5	FND507*	$1.50
8S3483	.5	727-Dual**	$2.50
8S2485	.5	727-Dual*	$2.50
8S2256	.6	747*	$1.95

*Common Anode **Common Cathode

POLY PAKS "CHIPS" AWAY IC AND CRYSTAL PRICES!

Order by Cat. No. 8S4048 and Type No.

Type	Description	Sale
ICM7205	Stopwatch	$14.95
AY3-8500-1	6 TV Games	9.95
AY5-9100	Telephone Dialer	12.95
AY5-9200	Repertory Dialer	9.95
MM5330	4½ Digit DVM	4.95
8038	Volt Control Osc	4.95
KR2376-139	BCD Encoder ROM	6.95
KR2376-ST	ASCII Encoder Rom	14.95
11C90DC	650MHz Prescaler	14.95
95H90DC	350MHz Prescaler	8.88
MC14410	Touch Ton Chip	9.95
MK2002P	Char Gen. (sim 2513)	3.95
7207A/7208	Freq. Counter pair	24.95
ICM7107	3½ Digit DVM/DMM	12.50
MM5369	60Hz Timebase IC	1.95
3.579MHz	Timebase Crystal	1.95
1.00MHz	Touchtone Crystal	5.95
3.2768MHz	Stopwatch Crystal	4.95
ULN2300	Op Amp With Built-In SCR	1.19
RC4195	Dual 5 Volt Tracking VR	2.95
TVR2000	High Power 723 Volt Reg	.59

DIP SWITCHES

Cat. No.	Switches	Sale
8S 3668	2	$.77
8S 3669	3	.88
8S 3021	4	.99
8S 3671	6	1.29

EECO 10-POSITION BCD THUMBWHEEL SWITCH

1-2-4-8 BCD encoding! Positions labeled 0-7, F, L. (F coded for 8, L coded for 9). White numerals on a black background. Eeco 800 series. 1 1/4 x 1 1/4 x 1/2". Cat. No. 8S3846

3 for $4. $1.49

GIANT SALE! MICRO-MINI TOGGLE SWITCHES

- 3A, 125VAC contacts or better
- Complete with mounting hardware!

Cat. No.	Contacts	Sale
8S4036	SPDT	$1.29
8S5085	SPDT*	1.39

*Center off

LINEARS

Type	Sale	Type	Sale
LM301V,H	$.25	LM380N	1.19
LM307V,N,H	.25	LM381N	1.19
LM308V	.19	LM555V	.44
LM309K	.99	LM558V	.19
LM311V	.19	LM565N,H	.65
LM320T-6	1.19	LM567V	1.40
LM340T-5	1.19	LM703H	.39
LM340T-12	1.19	LM709H	.19
LM340T-15	1.19	LM710N	.29
LM340K-5	1.19	LM741V,H	.29
LM340K-12	1.19	LM747H	.59
LM340K-15	1.19	LM1458V	.29
LM340K-18	1.19	LM1800N	.19
LM340K-24	1.19	LM3900N	.19
LM322N	1.19	LM3909V	.89
LM324N	.99	75491	.69
LM339N	1.09	75492	.75
LM377N	1.50	PA263	1.50

COMPUTER GRAPHICS "JOYSTICK" $4.50

2 for $8.

- Four 100K pots!
- Chrome handle and knob!

Wt. 10 oz. Cat. No. 8S3808A

IC SOCKETS

as low as 17¢ ea.

- Low profile, solder tail.

Cat. No.	Description	Sale
8S2123	8 pin minidip	$ .17
8S1308	14 pin dip	.19
8S1309	16 pin dip	.22
8S3378	18 pin dip	.25
8S5265	14 pin Wire Wrap*	.29
8S5266	16 pin Wire Wrap*	.35
8S3886	24 pin MSI	.59
8S3887	28 pin MSI	.59
8S3802	40 pin LSI	.99

*Gold Plated

RIBBON CABLE AT THIN PRICES

- Ultra-flat!
- 28 AWG!
- Single color! Indexed!

Cond.	Sale
20	8 ft. $1.98
26	6 ft. $1.98
34	5 ft. $1.98
40	4 ft. $1.98

Order by Cat. No. 8S3939 and conductors

SPECTRA-TWIST RIBBON CABLE

- Twisted pairs of brightly colored cable! 24AWG

Cat. No. 8S3680 48 cond. 2 ft. $1.98



MOTHERBOARD EDGE CONNECTOR

$2.50

106 pins (53 each side). Use with IMSAI & ALTAIR! 0.125" pin ctrs, goldplated 5/8" wire wrap leads. Open ends, fit wide PC boards. 8 ozs. Cat. No. 8S3987

IMC "PEWEE" BOXER FAN

$9.95

- Smallest Box Fan Made!
- Only 3 1/8" SQ, 1 5/8" Deep!

Cat. No. 8S5249

Quiet! Dependable! Powerful! Delivers 36 cfm Scarcely Audible Noise Level! Sleeve bearings rated for 10 years life! Perfect for cooling, ventillation, heating: for computers, power supplies, hi-fi eqpt, light projectors & more. 7-Blade flame-retardant polycarbonate fan, housed in die cast aluminum block. Wt. 1 lb.

For Hi-Fi and Computers. 115 VAC!

10 AMP POWER TAB SCR'S, TRIACS, QUADRACS!

Order by Cat. No. and Voltage

	Cat. No.
SCR'S	8S1730
TRIACS	8S1448
QUADRACS	8S1590

PIV	SALE
50	$.55
100	.69
200	.88

TRANSFORMER SALE

LOOK!

- 110VAC Primaries

Cat. No.	Output V.	Amps		Sale Each
8S 3399	6.3V	500ma	Metal encased	$1.98
8S 3814	6.3V	1A	Open frame	$2.49
8S 3412	12V	300ma	Open frame	$1.95

1N4000 Epoxy Rectifiers

- Miniature!
- 1.5 Amp!

Order by Cat. No. and Type No.

Cat. No.	Type No.	Volts	Sale	
8S2377	1N4001	50	10 for	$.75
8S2378	1N4002	100	10 for	.85
8S2379	1N4003	200	10 for	.95
8S2380	1N4004	400	10 for	1.19
8S2381	1N4005	600	10 for	1.39
8S2382	1N4006	800	10 for	1.49
8S2383	1N4007	1000	10 for	1.59

WRITE FOR POLY PAKS 1978 CATALOG

FREE!

FEATURING:

- Computer Components
- Over 400 Assortments!
- Digital Clocks, DPM's, and Stopwatches!
- Hi-Fi Stereos-Speakers!
- Solar Energy! Semi's!
- Test Equipment!
- Fiber Optics! Plus More n' More!

Terms: Add postage Rated: net 30
Phone: Wakefield, Mass. (617) 245-3829
Retail: 16-18 Del Carmine St., Wakefield,

MINIMUM ORDER — $6.00

POLY PAKS®

C.O.D.'s MAY BE PHONED

P.O. BOX 942-S8
LYNNFIELD, MA.
ZIP 01940

WE SHIP FROM STOCK

TELETYPE MODEL 43

Only $985

300 baud, full ASCII character set, 132 column width, 30 cps, quiet, light-weight, dependable. With RS 232 .... $1085

TECHNICO 9900 16 BIT COMPUTER KIT	$299
RCA VIP KIT	$230
EX-150 ELECTRONICS EXPERIMENT KIT	$65
HAZELTINE 1500	$1045
HAZELTINE 1500 KIT	$895
HAZELTINE 1510	$1165

To order: Check, money order, major credit cards. $4 shipping. NYC residents add 8% tax. We have no reader inquiry number. Please write or phone.

OWENS ASSOCIATES, DEPT. B-A
147 NORWOOD AVENUE
STATEN ISLAND, NEW YORK 10304

Day, Evening, Weekend Calls Welcome!

(212) 448-6283 (212) 448-6298

Apple II Computers

available at

Computer Components of Orange County

Available Now
Bob Bishop's Speech Synthesizer Program for APPLE II
... only $19.95
no special hardware needed

- Send for our Software list.
- Sign up for free Apple II workshops.



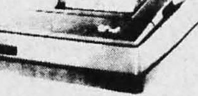
6791 Westminster Ave.
Westminster, CA 92683
714-898-8330

Circle 64 on inquiry card.

NEW SURPLUS PRINTER

Okidata – CP 110 – Friction Feed
Un-Used, in O.E.M. Factory Cartons.

- 5x7 Impact Dot Matrix
- 80 Char/Line
- 64 Char ASCII
- 110 Char/Sec.
- 66 Lines/Min.
- Accepts 8½" Roll paper



Includes – Power Supply, Built in Selftest, Parallel Interface, Line Buffer and Cables. Housed in a three piece plastic cabinet with all control electronics. Retail for over $1,100

Our Price: 1 ea. $650 10 up $600 ea.
Operating Manual Included.
Service Manual $20 *Supplies Limited*

Shipped Freight Collect. Send Check or M.O. Sorry No Warranty Available at these Prices

INTERNATIONAL ELECTRONICS EQUIPMENT CORP.
P.O. Box 522542, Miami, Florida 33152

Circle 172 on inquiry card.

IS YOUR COMPUTER DULL?

Is he hung up in the same old loop?
Is he sorting and searching his life away?
Turn him on to computer humor!
Bring out his hidden talents with a sophisticated selection of **BAUDY JOKES AND BYTING WIT!**
Let your computer comic delight your friends with a dazzling display of wit and whimsy! Your computer will have the last laugh!
Shift $5.95 into our register for:
LAF-TRAK 1 (on cassette)

Micro-Madness
P. O. Box 2250
Pasadena, Ca. 91105

Specify computer or interface type and memory size.

Circle 223 on inquiry card.

KIM KOMPUTER HEADQUARTERS
- Send for free KIM Katalog.

PAL COMPUTERS USING 6502 SYSTEM!
- ORDER OUR PAL TECHNICAL MANUAL ($7.25 Postpaid) which fully describes the KIM Bus and individual Boards in the PAL system.

INDUSTRIAL QUALITY MICROPROCESSOR CARD KITS
All +5V only, std 4.5 x 6.5, 22/44 edge conn.

4 K RAM	$79.95	2 K PROM	$ 79.95
Motherboard	$39.95	4 K PROM	$129.95
Card Cage	$20.00	Digital I/O	$ 59.95
Keyboard/Display	$89.95	Cassette Interface	$ 37.95

- 6802/6502 CPU BOARD $99.95 (SHOWN BELOW)

- BARE BOARDS, ENCLOSURES, FIRMWARE, ETC. SEND $3.70 FOR TECHNICAL MANUAL (INFORMATION ON ALL ABOVE INDUSTRIAL BOARDS)

SILVER SPUR
Electronic Communications Co.
P. O. Box 365, Chino, CA 91710
or Visit Retail Store:
13552 Central Avenue

- (714) 627-9366
- VISA
- Mastercharge
- Calif Residents Add 6% Sales Tax

Circle 317 on inquiry card.

PET SCHEMATICS

Another First From "PET-SHACK".

For only $34.95 you get:

24" x 30" schematic of the CPU board, plus oversized schematics of the Video Monitor and Tape Recorder, plus complete Parts Layout—all Accurately and Painstakingly drawn to the Minutest detail.

PET ROM ROUTINES

Another Breakthrough From "PET-SHACK".

Complete Assembly Listings of all 7 ROMs, plus identified subroutine entry points; Video Monitor, Keyboard Routine, Tape Record and Playback Routine, Real Time Clock, etc.

To entice you we are also including our own Machine Language Monitor program for your PET using the keyboard and video display. All this for only $19.95, *SATISFACTION GUARANTEED.*

You can have the Monitor program on cassette for only $9.95 extra.

Send check, money order, or Master Charge or VISA number and expiration date to:

PET-SHACK Software House
Marketing and Research Co.
P. O. Box 966
Mishawaka, IN 46544

Circle 289 on inquiry card.

IT'S A GREAT BIG COMPUTER WORLD
But You Only Need
THE COMPUTER CORNER

- **SOL** – A New Dawn is Here!
- **COMMODORE PET & KIM**
- **NORTHSTAR HORIZON**
- **IMSAI VDP-80**
- Memories & I/O Boards
- Computer Book Service
- Magnetic Tapes & Disks
- Full Line of Magazines
- Brain Games & Puzzles
- Workshops & Club Information

Visit **THE COMPUTER CORNER** for all your computer needs. Stop in and browse – you'll like our personal service.

THE COMPUTER CORNER
White Plains Mall – Upper Level
200 Hamilton Avenue
White Plains, New York 10601
Tel: (914) WHY - DATA

Ample Parking
10-6 Daily & Saturday
10-9 Thursday

Circle 65 on inquiry card.

S&D

SALE SALE SALE SALE SALE SALE
SUPPLIESSYSTEMS**SOFTWARE**PERIPHERALS**

8" FLEXIBLE DISKS	
53428 IBM 3740's, S/32, Shugart, Per Sci Pertec, Calcomp (send for catalog)	$ 3.90
53802 32 "Hard Sector", Altair, Shugart, Pertec, Per Sci, Calcomp & more	$ 3.90
5¼" FLEXIBLE DISKS	
54256 Single (01) Sector	$ 3.75
54257 Ten (10) Sector	$ 3.75
54258 Sixteen (16) Sector	$ 3.75
TELETYPE MODEL 43	
4320AAA Keyboard, Pin Feed, TTL	$1000.00
4310AAA Receive only, Pin Feed, TTL	$ 900.00
TP430035 Model 43 Ribbon	$ 5.25
PP43-A01 Model 43 Paper pin or fric, 3M	$ 12.75
SBCS1000 BUSINESS SYSTEM: microcomputer 32K 2 disk drives, video terminal, printer, cabinet/desk, DOS, BASIC, SOFTWARE	$6000.00
IECC1000 EDUCATOR SYSTEM: microcomputer 32K, disk drive, video terminal, printer cabinet/desk, DOS, BASIC, SOFTWARE	$5600.00

S & D COMPUTER TECHNOLOGY
P.O. Box 1283
ALBANY, NY 12201

24 HOURS/7 DAYS
(518) 463-2091

S&D

Circle 309 on inquiry card.

TRS 80

The Computer Pantry
presents

RECIPES (programs) on **CASSETTE TAPES** for your hungry **TRS 80**—with lots of graphics!!

All programs written to run on your **TRS 80—Level I** (4k or more RAM).

Free **COOKBOOK** (catalog) of cassettes with other recipes (programs) included.

CASSETTE 8010: U.F.O. Battle • Checkbook • Draw Poker • Pile High • Poker Dice • Snoopy (graphic).

CASSETTE 8020: Monthly Minder • Slot Machine • Dart Throw • Venus Touchdown • Number Guess • Tic Tac Toe.

$ 9.50 per cassette, $18.00 for both.

Send Check or Money Order to

THE COMPUTER PANTRY
2224-C Old Middlefield Way
Mountain View, CA 94043
(415) 964-4175

The Pantry pays postage and tax!! Prompt Delivery

Circle 77 on inquiry card.

Circle 353 on inquiry card.



$199.95
AS IS

$249.95
WORKING & CLEANED

IBM® Selectric-Based I/O Writers Excellent Hobby Printers

Series 72/731	All Solenoids
Heavy Duty	Original Documentation
8½" Platten	While Supply Lasts

SUPER SALE

These terminals are from a large airline reservation system. They are heavy duty and were under continuous maintenance. The units have been in storage. We make every effort to ensure that all essential parts are included. Most work when plugged in. No warranties are given or implied.

Conversion Kits.

1. Conversion instructons, P C board for printer only using software approach $59.95
2. I/O kit makes the unit into a conversational terminal. Instructions, P C board, components for a parallel or RS232 interface. Will work with any IBM terminal $249.95
3. Completely converted unit and assembled interface for I/O using kit $999.95

Card reader by HP with RS232 interface	$299.95
Printec line printers, parallel interface	$1500.00
Honeywell 516 & 316 mini's, make offer. Cables, used 11 conductor, 100 ft with connectors	$9.99
Electronic parts and circuit boards 1/4 lb bag	$4.99

Check, Money Order, Cash. Personal checks require 3 weeks to clear. No COD's. Units shipped UPS or PP collect. Prices Net FOB Tulsa

SUPER SURPLUS SALES
P.O. BOX 45944 TULSA, OK 74145 1-918-622-1058

4801 STATIC, TTL IN/OUT 4096x1 N-MOS RAM

GENERAL DESCRIPTION



Part Number 4801 is a 4K semiconductor random access memory organized as 4096 1-bit words. It is fully static and needs no clock or refresh pulses. It requires a single +5 volt power supply and is fully TTL compatible on input and output lines. The 4801 is packaged in a convenient 18 pin dual-in-line package.

FEATURES

- Single +5V Power Supply
- 4Kx1 Organization
- Replaces 4 1024x1 Static RAMs
- Completely Static—No Clocks or Refresh
- 18 Pin Package
- Access/Cycle Times 600 nsec max
- 250 mw Typical Operating Power
- Separate Data In and Data Out
- TTL Compatible I/O
- Three State Outputs
- Data Bus Compatible I/O Function

4804 STATIC, TTL IN/OUT 1024x4 N-MOS RAM

GENERAL DESCRIPTION



Part Number 4804 is a 4K semiconductor random access memory organized as 1024 4-bit words. It is fully static and needs no clock or refresh pulses. It requires a single +5 volt power supply and is fully TTL compatible on input and output lines. The 4804 is packaged in a convenient 18 pin dual-in-line package.

FEATURES

- Single +5V Power Supply
- 1Kx4 Organization
- Replaces 4 1024x1 Static RAMs
- Completely Static—No Clocks or Refresh
- 18 Pin Package
- Access/Cycle Times 600 nsec max
- 250 mw Typical Operating Power
- Common I/O Bus
- TTL Compatible I/O
- Three State Outputs

4801 or 4804 4K RAM's
$8.95 8/$60.00 16/$100.00

TRI-TEK
7808 North 27th Avenue
Phoenix, Arizona 85021
(602) 995-9352

10 AMP REGULATORS 78P05

GENERAL DESCRIPTION — The μA78P05 is a 3-terminal positive 5 V hybrid regulator capable of delivering 10 A. This device is virtually blowout proof and contains all the protection features inherent in monolithic regulators such as internal short-circuit current limiting and thermal-overload protection. The μA78P05 is packaged in a hermetically sealed TO-3 providing 50 W at 25°C case. The hybrid consists of a monolithic control chip driving a rugged Mesa transistor.

The high output current is achieved through new design technique without sacrificing the regulation characteristics of the controlling elements. The same process is employed in the construction of the 10 A regulator to provide the same high reliability obtained in the μA78H05 5 A regulator.

- 10 A OUTPUT CURRENT
- INTERNAL THERMAL-OVERLOAD PROTECTION
- INTERNAL SHORT-CIRCUIT CURRENT LIMIT
- LOW DROP-OUT VOLTAGE 2.2 V AT 10 A
- 50 W POWER DISSIPATION
- PIN-FOR-PIN COMPATIBLE WITH THE μA78H05, μA78H05A AND SH323
- STEEL TO-3 PACKAGE

CONNECTION DIAGRAMS
TO-3 PACKAGE
(TOP VIEW)
COMMON (3) OUTPUT (2) INPUT (1)

78P05SC..$12.95
Specs.. .60

HI POWER RECHARGEABLE CELL
Brand new Gates gelled electrolyte lead-acid energy cell. 2 Volts, 5 Amp Hour! Make high power stand-by supply for portable gear or computer systems.
GGC-0205..............................$5.35, 4/$27.00

MULTI-CHANNEL 8 BIT A/D CONVERTER
Fairchilds' new 6 channel analog-to digital converter has a lot going for it. Full scale correction capabilities, ratiometric conversion and wide input dynamic range.
Micro-processor compatible, it combines the multiplexer, decoder and sample-and-hold functions with converter to save board space and eliminate external parts. It provides 8 bit, $\pm\frac{1}{2}$ LSB conversion in 300 uSec featuring auto-zero and dynamic range all the way to ground.
UA9708 in 16 pin plastic DIP..........................$7.95!

- Please give street address for UPS shipping when possible.
- C.O.D. NO parcel post C.O.D. UPS C.O.D. Add 85¢ to order.
- Any correspondence not connected with your order, please use separate sheet and include SASE for reply.
- Orders less than $10 ($15 foreign) please add $1 handling.
- Prices are subject to change without notice.
- Any refunds will be by check, not credit vouchers.
- Terms. Check, money order, credit card. Net 30 days to rated firms, schools and government agencies.
- If we should be temporarily out of stock on an item, it will be placed on back order. If we cannot ship in 30 days, you will be notified of the expected shipping date and furnished with a postage paid card with which to cancel your order if desired.
- We pay surface shipping only in USA, Canada and Mexico.
- For premium shipping (first class, special handling, etc.) add extra. Excess will be refunded.
- Foreign orders (except Canada and Mexico) estimate and add shipping. Excess will be refunded.

INTEGRATED TONE RECEIVER MK5102(N)-5

FEATURES

- Detects all 16 standard DTMF digits
- Requires minimum external parts count for minimum system cost
- Uses inexpensive 3.579545 MHz crystal for reference
- Digital counter detection with period averaging insures minimum false response
- 16-pin package for high system density
- Single supply 5 Volts ± 10%
- Output in either 4-bit binary code or dual 2-bit row/column code
- Latched outputs

DESCRIPTION

The MK5102 is a monolithic integrated circuit fabricated using the complementary-symmetry MOS (CMOS) process. Using an inexpensive 3.579545 MHz television colorburst crystal for reference, the MK5102 detects and decodes the 8 standard DTMF frequencies used in telephone dialing. The requirement of only a single supply and its construction in a 16-pin package make the MK5102 ideal for applications requiring minimum size and external parts count.

DETECTION FREQUENCY

Low Group f_o	High Group f_o
Row 1 = 697 Hz	Column 1 = 1209 Hz
Row 2 = 770 Hz	Column 2 = 1336 Hz
Row 3 = 852 Hz	Column 3 = 1477 Hz
Row 4 = 941 Hz	Column 4 = 1633 Hz

MK5102N-5..............................$34.95
Specs...................................... .50

600 Ohm to 600 Ohm C.T. transformer.........$1.95

P. C. BOARD TERMINAL STRIP
Molded body encloses postive screw activated clamp which will accomodate wire sizes 14-30 AWG. Contacts and pins are solder plated copper. Pins are on .200 inch (5.08mM) for standard P. C. mounting. 10Amp rating. Compare our prices before you buy.

4 pole	TS-2504	.99
8 pole	TS-2508	1.49
12 pole	TS-2512	2.19

Circle 381 on inquiry card.

Unclassified Ads

MUST SELL: Heathkit H-8 & H-9 Terminal, assembled and operational. H-8: 16 K programmable memory, serial-cassette IO, cables, case & power supply. H-9: 12 x 80 characters, full cursor control, etc. Includes all software and manuals. Sell together or consider separately. $1400, shipping paid in US. Charles W Wood, 215 N Carswell, Whiteman AFB MO 65305, (816) 563-5094 nights or weekends.

WANTED: Printer, CRT terminal, and digital cassette recorder. Printer should have upper and lower case, ASCII code, RS-232 interface, typewriter quality output such as Diablo, IBM, Qume (no dot matrix), with or without keyboard; CRT Terminal with 80 character line width, block mode transmit, upper and lower case, RS-232; digital cassette recorder capable of working with printer or CRT terminal. Please indicate price, operating features, and condition. Norman N Axelrod, 445 E 86th St, New York NY 10028.

FOR SALE: New Altair 8800b computer. Includes 16 K x 8 memory, audio cassette record interface, serial 2SIO with 1 port, ASCII keyboard. Package II assembler, editor, debug and 8 K BASIC software by MITS, manuals. Ready to run. $1200 complete. TI 743RO 30 CPS printer, $1000. Ken Crandall, (415) 692-6693.

FOR SALE: Friden Flexowriter with attached 8 level tape punch and tape reader. The Flexowriter and punch both have independent code and binary modes. The Flexowriter has a row of control switches which include single step read, regeneration, delete and stop codes. The shift key shifts to a special character set which includes square root, delta, rub out and others. The Flexowriter has eight resettable tabs and is in good working condition. The schematics, modification sheets, specification sheets, code chart and metal table are all included. $200 or best offer plus shipping. Will trade. Anthony Masterson, 3812 Stratford, Dallas TX 75205.

FOR SALE: 16 K M6800 system of SwTPC boards and AC30 cassette interface and surplus keyboard. Includes Microware 16 task ROM, ACIA and PIA, editor, assembler, 8 K BASIC and full documentation. Excellent appearance and working order. Will sacrifice for $600 and deliver in New England. Robert Boyd, Woodlawn Av, Kennebunkport ME 04046.

WANTED: Complete documentation for VIATRON, Computer Tape Recorder (Model 5002 9 track). Must include schematics and parts list for unit and model 129000-00 power supply. R M Shectman, 14606 Larchmere Blvd, Shaker Hts OH 44120.

FOR SALE: Ohio Scientific OSI 430B Super IO board, fully populated; A to D, D to A, Multiplex, cassette interface (1200 baud KC) serial IO parallel IO etc. Brand new, all ICs in sockets. Cost $399 new, sell for $275. Joe Ruby, 6057 Beck Av, N Hollywood CA 91606, (213) 985-1519.

NEW UNCLASSIFIED POLICY

Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed notice to that effect. To be considered for publication, an advertisement must be clearly noncommercial, typed double spaced on plain white paper, contain 50 words or less, and include complete name and address information.

These notices are free of charge and will be printed one time only on a space available basis. Notices can be accepted from individuals or bona fide computer users clubs only. We can engage in no correspondence on these and your confirmation of placement is appearance in an issue of BYTE.

Please note that it may take three or four months for an ad to appear in the magazine.■

FOR SALE: Motorola 6800 based d-2 MPU kit complete in unopened package. Includes keyboard and display, monitor program, 378 bytes programmable memory, serial and parallel IO ports. Everything you need except the 5 V 2 A power supply to get started in MPUs. Sell for $170, kit lists for $235. Steve Kay, 51 Tidewater Ln, Willingbor NJ 08046, (609) 877-1345.

SACRIFICE: 8 K 8080A system six parallel IO ports (add two chips for three more PIOs and serial), 3 K PROM board (w/o 2708), 2-4 K 450 ns low power static programmable memory, TVT-3 terminal with cursor control, UART brd, 2 K programmable memory board, Hall effect ASCII keyboard, HITS audio cassette interface. All boards new, unused, sockets for chips, most boards assembled. 5 A supply, edge connectors and card guides (extra included), case, documentation. Bonus 2-60 chip ground plane wire wrap proto boards with 96 gold plated wire wrap sockets. Sacrifice at fraction of cost $675 (sent UPS). Write for information, David Krivoshik, 18 Newcomb, Elizabeth NJ 07202.

FOR SALE: Iomec disk drive with single removable 1 M byte platter, with power supply, rack and manual. Also: CDC 300 lines per minute drum printer with manual. No interfaces available. Machines sold as is, neither have been used for over 3 years. Asking $1,000 each. Chuck Gahan, 12781 Taylor St, Garden Grove CA 92645.

WANTED: Boards for 8008 computer and manual for Mark 8. I hope to find someone who never finished a Mark 8 or MOD 8 and will sell cheaply. Larry Bates, POB 664, US 25, Skyland NC 28776.

FOR SALE: Model 400 M6800 processor (OSI) with all documentation, buss holds 8 boards, 4 K memory board (3 K on board), and video board. All for $150 or best offer. Tony Preston, 15151 Ripple Dr, Linden MI 48451.

FOR SALE: Tarbell cassette interface board. Assembled, checked at factory, with manual. For price of kit only $120. E Tottle, 2993 Yorkway, Baltimore MD 21222.

FOR SALE: MITS Dual Disk System, 8800A with heavy duty power supply, dual disk drive, 40 K Seals memory, ADM-1 CRT, Okidata printer, 2SIO board, PROM board, disk boot loader on PROM, ACR board. Best offer. J R Morrow, 6501 Harding Rd, E21, Nashville TN 37205, (615) 329-1844 days, (615) 352-0599 evenings.

FOR SALE: Teletypes, ASR33s, one brand new in box with manuals, located in Silicon Valley, one used only 25 hours, located in Boston. $500 each or reasonable offer. If interested in either one, send postcard to Ralph Guditz, POB 61266, Sunnyvale CA 94088.

FOR SALE: Centronics 101A with lower case option, parallel input, switch or software for double width characters. Includes stand and paper tray. $1200, you pay freight. Digital Group 8 K memory boards 400 ns low power. Two at $200 each, I pay shipping. D L Earle, 729 Milam Bldg, San Antonio TX 78205, (512) 222-0077 or (512) 690-0499.

FOR SALE: Heath H9 Video Terminal. New and completely assembled, $500. Jim Ketcham, 57F Sherwood Forest, Wappingers Falls NY 12590, (914) 831-1709 evenings.

FOR SALE: One fully assembled and functioning TDL ZPU Z-80 processor card, $250. Includes operating manual and 1 K monitor. One fully assembled Seals battery backup card with batteries, $120. Contact James Stanley, #805, 4045 Linkwood, Houston TX 77025, or call (713) 792-4727.

WE NEED HELP! High school has a Sol System II with 32 K and a video tape camera and cassette. We would like to interface the two in order to produce digitized computer portraits. Send information or suggestions to Bill Games, Franklin HS, Stockton CA 95205.

FOR SALE: RCA COSMAC VIP. Described in August 1977 BYTE, page 30. Original cost $275 in kit form. Will sell for $200 complete in running order. Albert G Shafer, 683 SW 7th St, Boca Raton FL 33432, (305) 395-5633.

FOR SALE: BYTE volume 1, #1 to date, for best offer. Good condition. Will pack securely and ship insured. Also, complete Viatron System 21 with manuals. Includes 9 inch B&W and 19 inch color video display, robot printer, two cassette decks, all in good working order. George Weiss, 4625 N Kedzie Av, Chicago IL 60625, (312) 588-1885.

FOR SALE: Diablo Systems Model 31 disk drive with Interdata controller. Disk drive is 100 percent operational, controller needs some work. Full documentation supplied for both drive and controller. Best offer over $1200. Tom Olsen, 655 South Fairoaks Ave, Apt H206, Sunnyvale CA 94086, (408) 737-9353.

FOR SALE: Technical Design Labs Xitan Alpha 2 computer. Complete with ZPU, Z-16 (16 K static low power programmable memory), SMB (System monitor in ROM, 1200 bps cassette IO, two serial IO ports, one parallel IO port), mother board (eight slots, currently using three), power supply, card rack, fan, case. Assembled, tested, fully operational with 8 K BASIC, assembler, text editor and text output processor supplied on cassette. $1400 buys all, including Pioneer CT 4141 cassette deck. An excellent starter system. J Crowley, 2931 Queen Ln, Philadelphia PA 19129, (215) 842-1517.

FOR SALE: Assembled solid state music video display and graphics board $125. ($25 under kit price!) Extender card with connector $9. Six slot mother board (can be connected to other board for expansion) with 4 connector, $25. Good for small expandable system. Send SASE to Gary Camp, 158 Pago Pago, Pomona CA 91768.

FOR SALE: TELETYPE Model 33 ASR, with pedestal and GE acoustic coupler assembly. Complete $400. Full set manuals available. Porta-com Model PC8110, standard ASCII complete, with internal acoustic coupler and manual cost $900, selling for $400. Ron Newland, 206 S Highwood Av, Glen Rock NJ 07452, (201) 444-6872.

FOR SALE: Sylvanhills DFT-2 flat bed plotter with power supply, 17 by 22 inch drawing area, mounted on drafting table, could be shipped unmounted. Also a Matrix video S-100 bus board ALT 256-2, high density graphics with a software package for character generation. All items almost new. Laddie Chapman, 5715 Klump Av, N Hollywood CA 91601, (213) 985-5715.

FOR SALE: TRS-80 Microprocessor with tape unit and RF modulator for use with any TV. Brand new, still under factory warranty. Only $499 plus shipping. Reason for sale: moving up to more powerful system. Write R Lufkin, Star Rt 1, POB 4, Charlottesville VA 22901.

MEDICAL SOFTWARE FOR MICROPROCESSORS: Now available, patient history, health risk profile, differential diagnosis generators, clinical laboratory data analysis, symptom, disease and drug cross-indexing, bibliography-reference retrieval and others. Write R Lufkin, Star Rt 1, POB 4, Charlottesville VA 22901.

HELP: I couldn't resist the price so I bought it. Model-140 Eldorado computer, 8 K core. Selectric typewriter. Desk console. Three magnetic tape cassette drives. Programs and compilers. Includes a whole business accounting package. I need to find schematics in case it breaks. G Kilthau, 7142 Tours, Houston TX 77036.

FOR SALE: 2 Digital Group 8 K memory boards. Uses 2102-1 chips. Professionally assembled and running perfectly for three months. $200 each. L Langrehr, 2069 N Humboldt Blvd, Chicago IL 60647, or call (312) 262-1600 X491.

VISIBLE OR INFRA RED

USED FOR CHARACTER RECOGNITION FOR COMPUTERS WITH EXTERNAL CIRCUITS

MAY BE USED IN A VACUUM, UNDER WATER, HIGH ALTITUDE

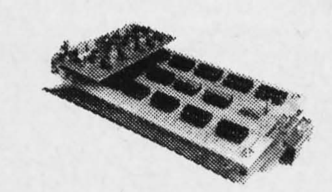
IN MAGNETIC ENVIRONMENT BECAUSE THERE IS NO HIGH VOLTAGE OR MAGNETIC DEFLECTION

MINATURE SOLID STATE

202 VIDEO CAMERA KIT

FEATURING A. . .100 × 100 BIT SELF SCANNING CHARGED COUPLED DEVICE

THIS UNIQUE UPDATED CAMERA KIT FEATURES THE FAIRCHILD CCD 202C IMAGE SENSOR

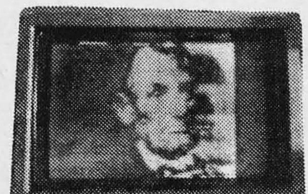


ADVANTAGES

- **IN THE FUTURE WE WILL SUPPLY A COMPUTER VIDEO INTERFACE CARD**
- All clock voltages operate at 6V reguiring no adjustments
- Higher video output signal
- We supply the power board, so only a 5V 1 Amp power source is needed
- The circuitry has been simplified for easier assembly
- Two level TTL output is supplied for interfacing

FEATURES

- Sensitive to infra red as well as visible light
- May be used for IR surveillance with an IR light source
- Excellent for standard surveillance work, because of light weight and small size
- All components mounted on parallel 3¾" × 6½" single sided boards
- Total weight under 1 lb.



$349.00 KIT

Add $75.00 to assemble and test

Add $2.00 Postage and Handling

We supply all semiconductors, boards, data sheets, diagrams, resistors and capacitors, and 8MM lens.
Sorry we do not supply the case, batteries and 5V supply.

SURPLUS CENTRONICS PRINTERS

101 — $1,150

306 — $900

UNIVERSAL 4Kx8 MEMORY BOARD KIT

$69.95

32-2102-1 fully buffered, 16 address lines, on board decoding for any 4 of 64 pages, standard 44 pin buss, may be used with F-8 & KIM

EXPANDABLE F8 CPU BOARD KIT

$99.00

featuring Fairbug PSU.1K-of static ram, RS 232 interface, documentation, 64 BYTE regis.er

C/MOS (DIODE CLAMPED)

4001 –	.18	4016 –	.29	4027 –	.37	4053 –	1.10
4002 –	.18	4017 –	.90	4028 –	.80	4055 –	1.25
4006 –	.95	4018 –	.90	4029 –	.95	4066-	.70
4007 –	.18	4019 –	.37	4030 –	.33	4071 –	.18
4009 –	.37	4020 –	.90	4035 –	.97	4076 –	.97
4010 –	.37	4021 –	.90	4042 –	.65	4518 –	1.00
4011 –	.18	4022 –	.90	4046 –	1.35	74C10	.22
4012 –	.18	4023 –	.18	4047 –	1.50	74C193	.95
4013 –	.29	4024 –	.75	4049 –	.34		
4015 –	.74	4025 –	.18	4050 –	.34		

2708-8K EPROM (450 ns) .... $ 9.75
2522 STATIC SHIFT RE6 .... $ 1.95
2513 CHARACTER GEN .... $ 6.75
2518 HEX 32 BIT SR .... $ 2.20
2102-1 (450ns) .... $ .99
21LO2-1 (450ns) .... $ 1.25
MM5270-4KXI DYN .... $ 3.45
MK 4008P .... $ 1.95
2101-1-254 x 4 STATIC .... $ 2.45
2111-1-256 x 4 STATIC .... $ 3.45
2112-1-256 x 4 STATIC .... $ 2.75
2114-4K STATIC RAM (450ns) .... $ 8.50
5280/2107B 4K DYNAMIC RAM .... $ 4.25
TMS 4050L .... $ 3.95
5204-4K PROM .... $ 4.95
82S23 .... $ 1.95
AY-5-1013 UART .... $ 4.95
8703C TELEDYNE 3 STAGE BINARY 8 BIT A/D CONVERTER $13.50
8080A .... $ 9.50

AP BOARDS USED FOR SOLDERLESS BREADBOARDING

264L — 128 FIVE-TIE POINT TERMINALS $12.50
212R — POWER BOARD .... $ 2.50
2248L — 96 FIVE-TIE POINT TERMINALS $10.00
209R — POWER BOARD .... $ 2.25

IC TEST CLIPS

TC-14 — $4.50 TC-16 — $4.75

CTS 206-8 eight position dip switch $1.60
CTS-206-4 four position dip switch $1.45
LIGHT ACTIVATED SCR's TO-18, 200V 1A. .... $ 1.10

SILICON SOLAR CELLS

2¼" diameter .4V at 500 ma $4.00

LED READOUTS

FND 359 C.C. .4" $ .50
FCS 8024 · 4 digit C.C. 8" display $5.95
FND 503 C.C. .5" $ .85
FND 510 C.A. .5" $ .85
DL-704-.3" C.C. $ .75
DL-704 C.A. .3" $ .75
DL 747 C.A. .6" $1.25
FND 800 C.C. .8" $1.95
FND 807 C.A. .8" $1.95

PRINTED CIRCUIT BOARD

4-1/2"x6-1/2" SINGLE SIDED EPOXY BOARD 1/16" thick, unetched $.60 ea. .... 5/$2.60

7 WATT LD-65 LASER DIODE IR $8.95

2N 3820 P FET .... $ .45
2N 5457 N FET .... $ .45
2N2646 UJT .... $ .45
ER 900 TRIGGER DIODES .... 4/$1.00
2N 6028 PROG. UJT .... $ .65

MINIATURE MULTI-TURN TRIM POTS 100, 1K, 2K, 5K, 10K, 25K, 50K, 100K, 200K, 500K, 1Meg, 2Meg, $.75 each 3/$2.00

CHARGED COUPLE DEVICES

CCD 321 AH2-512 audio analog S.R. $95.00
CCD 201C 100x100 Image Sensor $95.00
CCD 202C 100x100 Image Sensor $145.00

VERIPAX PC BOARD $4.00
This board is a 1/16" single sided paper epoxy board, 4½"x6½" DRILLED and ETCHED which will hold up to 21 single 14 pin IC's or 8,16 or LSI DIP IC's with busses for power supply connector.

FP 100 PHOTO TRANS .... $ .50
RED, YELLOW, GREEN or AMBER LARGE LED's .2" .... 6/$1.00
TIL-118 OPTO-ISOLATOR .... $ .75
MOLEX PINS .... 100/$1.00 1000/$8.00
10 WATT ZENERS 3.9, 4.7, 5.6, 8.2, 18,22,100,150 or 200V. ea. $ .60
1 WATT ZENERS: 3.3, 4.7, 5.6, 9.1,10, 12, 15, 18, or 22V .... 7/$1.00
MC6860 MODEM CHIP $9.95
MCM 6571A 7 x 9 character gen ... $10.75

Silicon Power Rectifiers

PRV	1A	3A	12A	50A	125A	240A
100	.06	.14	.30	.80	3.70	5.00
200	.07	.20	.35	1.15	4.25	6.50
400	.09	.25	.50	1.40	6.50	9.50
600	.11	.30	.70	1.80	8.50	12.50
800	.15	.35	.90	2.30	10.50	16.50
1000	.20	.45	1.10	2.75	12.50	20.00

SAD 1024-a REDICON 1024 stage analog "Bucket Brigade" shift register. $18.95

IN 4148 (IN914) .... 15/$1.00

RS232 CONNECTORS

DB 25P male $2.95
DB 25S female $3.50

REGULATORS

309K .... $ .95
723 .... $ .50
LM 376 .... $ .60
320K-5, 1 or 15V ... $1.00
320T-5, 1 or 24V .... $ .95
340K-12,15 or 24V. ... $ .95
340T-5, 6, 8, 12 15,18 or 24V $ .95
78 MG $1.35
79 MG $1.35

TRANSISTOR SPECIALS

2N6233-NPN SWITCHING POWER $ 1.95
MRF-8004 a CB RF Transistor NPN $ 1.50
2N3772 NPN Si TO-3 .... $ 1.00
2N1546 PNP GE TO-3 .... $ .75
2N4908 PNP Si TO-3 .... $ 1.00
2N6056 NPN Si TO-3 Darlington .... $ 1.70
2N5086 PNP Si TO-92 .... 4/$ 1.00
2N3137 NPN Si RF .... $ .85
2N3919 NPN Si TO-3 RF .... $ 1.50
2N1420 NPN Si TO 5 .... 3/$ 1.00
2N3767 NPN Si TO-66 .... $ .70
2N2222 NPN Si TO-18 .... 5/$ 1.00
2N3055 NPN Si TO-3 .... $ .50
2N3904 NPN Si TO-92 .... 5/$ 1.00
2N3906 PNP Si TO-92 .... 5/$ 1.00
2N5296 NPN Si TO-220 .... $ .50
2N6109 PNP Si TO-220 .... $ .55
2N3638 PNP Si TO-5 .... 5/$ 1.00
MPSA 13 NPN Si. .... 4/$ 1.00

TTL IC SERIES

7400–	.13	7445–	.65	74151–	.61
7401–	.13	7446–	.68	74153–	.61
7402–	.13	7447–	.58	74154–	.94
7403–	.13	7448–	.68	74155–	.58
7404–	.15	7450–	.15	74157–	.55
7405–	.13	7472–	.25	74161–	.55
7406–	.16	7473–	.28	74163–	.55
7407–	.20	7474–	.28	74164–	.85
7408–	.18	7475–	.45	74165–	.95
7409–	.18	7476–	.30	74170–	1.68
7410–	.13	7480–	.31	74173–	1.20
7411–	.18	7483–	.65	74174–	.95
7412–	.13	7485–	.87	74175–	.85
7413–	.36	7486–	.28	74176–	.75
7414–	.60	7490–	.42	74177–	.75
7416–	.22	7491–	.58	74180–	.65
7417–	.25	7492–	.43	74181–	1.90
7420–	.13	7493–	.43	74190–	1.00
7425–	.25	7494–	.67	74191–	1.00
7426–	.22	7495–	.65	74192–	.79
7427–	.19	7496–	.65	74193–	.79
7430–	.13	74107–	.28	74194–	.80
7432–	.22	74121–	.29	75325–	1.50
7437–	.21	74122–	.38	74196–	.86
7438+	.21	74123–	.45	74279–	.55
7440–	.13	74125–	.40	74367–	.65
7441–	.70	74126–	.40	75491–	.50
7442–	.37	74150–	.94	75492–	.50

DATA CASSETTES $ .95

44 Pin Solder Tail .156" Connector $1.95

MM 5387AA new clock chip which will directly drive LED's 12/24 hrs., 1 supply & alarm $5.95

NO. 30 WIRE WRAP WIRE SINGLE STRAND 100' $1.40

ALCO MINIATURE TOGGLE SWITCHES

MTA 106 SPDT .... $ .95
MTA 206 DPDT .... $ 1.70
MTA 206 P-DPDT CENTER OFF $ 1.85
MSD 206 P-DPDT CENTER OFF LEVER SWITCH .... $ 1.85

Full Wave Bridges

PRV	2A	6A	25A
100			1.30
200	.75	1.25	2.00
400	.95	1.50	3.00
600	1.20	1.75	4.00

DIP SOCKETS

8 PIN .17 24 PIN .35
14 PIN .20 28 PIN .40
16 PIN .22 40 PIN .60
18 PIN .25

SANKEN AUDIO POWER AMPS

Si 1010 G 10 WATTS .... $ 7.80
Si 1020 G 20 WATTS $15.70
Si 1050 G 50 WATTS $28.50

TANTULUM CAPACITORS

.22UF 35V 5/$1.00	6.8UF 35V 4/$1.00
.47UF 35V 5/$1.00	10UF 10V $ .25
.68UF 35V 5/$1.00	22UF 25V $ .40
1UF 35V 5/$1.00	15UF 35V 3/$1.00
2.2 UF 20V 5/$1.00	30UF 6V 5/$1.00
3.3UF 35V 4/$1.00	47UF 20V $ .35
4.7UF 15V 5/$1.00	68 UF 15V $ .50

74LS SERIES

74LS00 –	.20	74LS132 –	.70
74LS02 –	.21	74LS138 –	.70
74LS03 –	.21	74LS139 –	.70
74LS04 –	.23	74LS151 –	.65
74LS05 –	.24	74LS153 –	.65
74LS08 –	.23	74LS155 –	.62
74LS09 –	.23	74LS156 –	.62
74LS10 –	.20	74LS157 –	.62
74LS11 –	.20	74LS160 –	.80
74LS13 –	.40	74LS161 –	.80
74LS14 –	.80	74LS162 –	.80
74LS15 –	.25	74LS163 –	.80
74LS20 –	.22	74LS168 –	.80
74LS21 –	.22	74LS169 –	.80
74LS22 –	.22	74LS170 –	1.50
74LS26 –	.30	74LS173 –	1.00
74LS27 –	.25	74LS174 –	.75
74LS28 –	.32	74LS175 –	.75
74LS30 –	.21	74LS181 –	2.00
74LS32 –	.32	74LS190 –	.90
74LS37 –	.30	74LS191 –	.90
74LS38 –	.30	74LS192 –	.90
74LS40 –	.25	74LS193 –	.90
74LS42 –	.60	74LS195 –	.50
74LS47 –	.73	74LS196 –	.75
74LS51 –	.25	74LS197 –	.75
74LS54 –	.25	74LS221 –	1.00
74LS73 –	.35	74LS257 –	.70
74LS74 –	.35	74LS258 –	.70
74LS75 –	.50	74LS265 –	.25
74LS76 –	.35	74LS279 –	.50
74LS85 –	1.00	74LS290 –	.60
74LS86 –	.35	74LS293 –	.60
74LS90 –	.50	74LS365 –	.52
74LS92 –	.50	74LS366 –	.52
74LS93 –	.50	74LS367 –	.52
74LS109 –	.35	74LS368 –	.52
74LS112 –	.35	74LS375 –	.60
74LS113 –	.35	74LS386 –	.35
74LS114 –	.35	74LS390 –	1.50
74LS125 –	.45	74LS670 –	2.00
74LS126 –	.45		

LINEAR CIRCUITS

LM 101 –	.75
LM 301/748	.25
LM307 –	.30
LM 308 –	.75
LM 311 –	.75
LM 318 –	1.20
LM 319 –	.95
LM 324 –	.70
LM 339 –	1.10
LM 358 –	.70
LM 370 –	1.15
LM 377 –	1.60
LM 380 –	.95
LM 381 –	1.25
LM 382 –	1.25
LM 537 –	2.50
LM 553 –	2.50
LM 555 –	.39
LM 556 –	.85
NE540L –	2.25
560 –	2.00
565	.95
566 –	1.10
567 –	1.10
703 –	.90
709 –	.25
710 –	.35
741C or V –	.25
747 –	.50
LM 1310 –	2.50
1456 –	.95
1458 –	.50
CA3046 –	.75
3900 –	.40
8038CC –	3.90
791 –	1.95
LF356H –	1.20

TRIACS / SCR'S

	TRIACS			SCR'S		
PRV	1A	10A	25A	1.5A	6A	35A
100	.40	.70	1.30	.40	.50	1.20
200	.70	1.10	1.75	.60	.70	1.60
400	1.10	1.60	2.60	1.00	1.20	2.20
600	1.70	2.30	3.60		1.50	3.00

Terms: FOB Cambridge, Mass. Send Check or Money Order. Include Postage, Minimum Order $5.00, COD'S $20.00

Send 25¢ for our catalog featuring Transistors and Rectifiers 145 Hampshire St., Cambridge, Mass.



SOLID STATE SALES
P.O. BOX 74B
SOMERVILLE, MASS. 02143 TEL. (617) 547-7053

WE SHIP OVER 95% OF OUR ORDERS THE DAY WE RECEIVE THEM

Reader Service

To get further information on the products advertised in BYTE, fill out the reader service card with your name and address. Then circle the appropriate numbers for the advertisers you select from the list. Add a 13 cent stamp to the card, then drop it in the mail. Not only do you gain information, but our advertisers are encouraged to use the marketplace provided by BYTE. This helps us bring you a bigger BYTE.

**Correspond directly with company.*

BOMB– BYTE's Ongoing Monitor Box

BOMB's Vacation

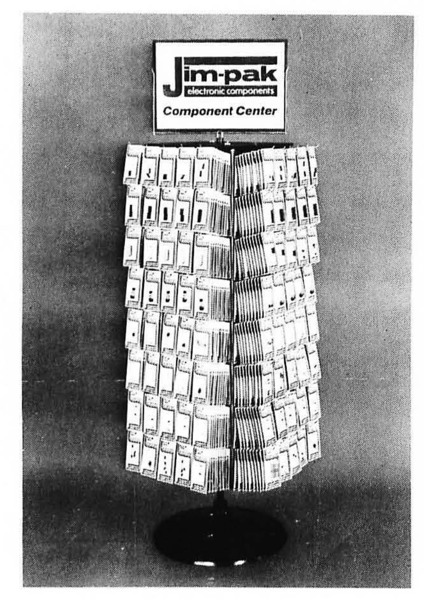
The Bomb Analysis for May 1978 BYTE found Mark Gottlieb's "Hidden Line Subroutines for Three-Dimensional Plotting," page 49, receiving the highest score, with Larry Weinstein's "A Programmable Character Generator, Part 1: Hardware," page 79, receiving second place. Since our pocket calculator's batteries had run down and the charger got misplaced, we'll omit the statistical analysis this month. . .CH■

Jim-pak® electronic components

One-Stop Component Center

AUTHORIZED DEALERS

ALABAMA	
Mobile	*Lafayette Radio Electronics*
ALASKA	
Anchorage	*Electronics Corp. of Alaska*
ARIZONA	
Flagstaff	*Jim's Audio & Stereo Repair*
Fountain Hills	*P & C Communications*
Tempe	*Computerworld Inc.*
Sierra Vista	*B & S Electronics*
Yuma	*Yuma Electronics*
CALIFORNIA	
Bellflower	*Earl's Hobby Shop*
Berkeley	*Al Lasher Electronics*
Brea	*Century Electronics*
Cypress	*SCR Electronics*
El Monte	*Kimball & Stark*
Fontana	*Fontana Electronics*
Fullerton	*Orvac Electronics Inc.*
Glendale	*Eagle Electronics*
Lake Tahoe, South	*CalPine Electronics*
Lancaster	*Consumer Electronics*
Long Beach	*Scott Radio Supply Inc.*
Mission Viejo	*Tower Electronics Corp.*
Modesto	*Computer Magic*
Monterey	*Zackit*
Oceanside	*Electronic Center*
Palmdale	*Radio Shack A.S.C. Palmdale*
Palo Alto	*Zack Electronics*
Pasadena	*Dow Radio Inc.*
Riverside	*Computer Center*
Sacramento	*Heathkit Electronic Center*
Sacramento	*The Radio Place*
Sacramento	*Zackit*
San Bernardino	*Inland Computer & Electronics*
San Carlos	*J & H Outlet Store*
San Diego	*Radio Shack A.S.C. Mira Mesa*
San Diego	*Radio-Tronics Inc.*
San Fernando	*San Fernando Electronics*
San Francisco	*Zack Electronics*
San Francisco	*Zenith Distributing Corp.*
San Jose	*Quement Electronics*
San Luis Obispo	*Mid-State Electronic Supply*
San Rafael	*Electronics Plus*
Santa Barbara	*Lombard Electronics*
Santa Cruz	*Santa Cruz Electronics*
Santa Maria	*Caps Electronics*
Santa Monica	*Mission Control*
Sunnyvale	*Sunnyvale Electronics*
Torrance	*SE Electronics*
Vallejo	*Zackit*
Van Nuys	*Thrifty Electronics Supply*
Ventura	*Lombard's Electronics Inc.*
Walnut Creek	*Byte Shop of Walnut Creek*
Westminster	*JK Electronics*
Whittier	*D & S Electronics*
COLORADO	
Aurora	*Com Co Electronics*
Steamboat Springs	*Norm's TV & Electronics*
CONNECTICUT	
Bridgeport	*Bridgeport Computer*
FLORIDA	
Ft. Lauderdale	*Computers For You*
Gainesville	*Lafayette Radio*
Lakeland	*Lakeland Specialty Electronics*
Orlando	*Altair Computer Center*
Tampa	*AMF Electronics*
Tampa	*Microcomputer Systems*
GEORGIA	
Atlanta	*Atlanta Computer Mart*
HAWAII	
Aiea	*Delcoms Hawaii*
Honolulu	*Integrated Circuit Supply*
IDAHO	
Idaho Falls	*Audiotronics*
ILLINOIS	
Carbondale	*Lafayette Radio*
Evanston	*Itty Bitty Machine Co.*
Evanston	*Tri-State Electronic Corp.*
Granite City	*Computer Systems Center*
Groveland	*Moyer Electronics*
Mount Prospect	*Tri-State Electronic Corp.*
Niles	*Computer Land*
Oak Park	*Spectronics Inc.*
Rockford	*Imperial Computer Systems*
Schaumburg	*Data Domain*
INDIANA	
East Chicago	*Acro Electronics Corp.*
Hammond	*Quantum Computer Works*
IOWA	
Clinton	*Bridge Elec. Computer Center*
Davenport	*Computer Store of Davenport*
Indianola	*Electronix Limited*
KANSAS	
Kansas City	*Electronic Surplus Sales*
KANSAS (Continued)	
Manhattan	*Communications Specialties Ltd.*
Wichita	*Amateur Radio Equipment Company*
KENTUCKY	
Lexington	*Radio-Electronic Equipment Co.*
LOUISIANA	
Baton Rouge	*Davis Electronics Supply Co.*
MARYLAND	
Baltimore	*Computer Workshop of Baltimore*
Baltimore	*Everything Electronic*
La Vale	*J & M Electronics*
Rockville	*Computer Workshop*
Silver Spring	*Computers Etc.*
Towson	*Baynesville Electronic Inc.*
Towson	*Computers Etc.*
MASSACHUSETTS	
Medford	*Tufts Electronics*
North Adams	*Electronics Supply Center*
Waltham	*Computer Mart Inc.*
Worcester	*RM Electronics Inc.*
MICHIGAN	
Ann Arbor	*Airway Electronic Communications*
Flint	*Hobby Electronic Center*
Grand Rapids	*Micro Computer World*
Lansing	*Fulton Radio Supply Co.*
Mt. Clemens	*The Computer Store*



MINNESOTA	
Duluth	*Northwest Radio of Duluth*
Eagan	*Computer Room Inc.*
Hopkins	*Heathkit Electronic Center*
MISSOURI	
El Dorado Springs	*Beckman Electronics*
Parkville	*Computer Workshop of Kansas City*
MONTANA	
Billings	*Conley Radio Supply*
NEBRASKA	
Lincoln	*Altair Computer Center*
North Platte	*Scott Radio Supply Corp.*
Omaha	*Omaha Computer Store*
NEVADA	
Las Vegas	*Century 23*
NEW JERSEY	
Bayville	*A.R.S. Communications Services*
Bricktown	*Radio Shack Associate Store*
Cherry Hill	*The Computer Emporium*
Hoboken	*Hoboken Computer Works*
Paterson	*All-tronics*
Pompton Lake	*Computer Corner of New Jersey*
Ramsey	*Typetronic Computer Store*
NEW YORK	
Albany	*Fort Orange Electronics*
Kingston	*Greylock Electronics*
New York	*Computer Mart of New York*
NEW YORK (Continued)	
Rensselaer	*Com-Tech Electronics*
Rochester	*2001 Microsystems*
Troy	*Trojan Electronics*
White Plains	*The Computer Corner*
Williamsville	*Hirsch Sales Co.*
NORTH CAROLINA	
Durham	*Futureworld*
Greensboro	*Byte Shop*
Raleigh	*Byte Shop of Raleigh*
OHIO	
Bucyrus	*Mead Electronics*
Cincinnati	*Digital Design*
Columbus	*Heathkit Electronic Center*
Dayton	*Altair Computer Center*
Reynoldsburg	*Universal Amateur Radio*
OKLAHOMA	
Guymon	*Sound Service*
Oklahoma City	*Bits, Bytes & Micros*
Tulsa	*High Technology*
OREGON	
Albany	*Oregon Ham Sales*
Beaverton	*Altair Computer Center*
Coos Bay	*Herrick Electronix*
Medford	*Portland Radio Supply*
Ontario	*Miller Electronics*
Portland	*Portland Radio Supply*
Salem	*Computer Pathways*
PENNSYLVANIA	
Drexel Hill	*Kass Electronic Distributors*
Erie	*Warren Radio*
Hershey	*Microcomputer Systems Inc.*
Murraysville	*Computer Workshop of Pittsburgh*
Phoenixville	*Stevens Electronics*
Pittsburgh	*Tydings Company*
Wilkesbare	*Hamline Electronics*
York	*G.Y.C. Company*
RHODE ISLAND	
Cranston	*Jabbour Electronics City*
Pawtucket	*Jabbour Electronics City*
SOUTH CAROLINA	
North Charleston	*Technical Services Inc.*
TENNESSEE	
Chattanooga	*William's Data Comp Division*
Clarksville	*Masstronics*
Knoxville	*Byte Shop*
Memphis	*Bluff City Electronics*
Memphis	*Sere-Rose & Spencer Electronics*
Oak Ridge	*Computer Denn*
TEXAS	
Amarillo	*Computer Encounters Inc.*
Dallas	*CompuShop*
Houston	*Altair Computer Center*
Houston	*Interactive Computers*
San Antonio	*Sherman Electronics Supply Inc.*
UTAH	
Provo	*Alpine Electronic Supply Co.*
VIRGINIA	
Alexandria	*Computer Hardware Store*
Alexandria	*Computers Plus Inc.*
Charlottesville	*Lafayette Electronics*
Hampton	*Lafayette Radio*
Richmond	*Computers-To-Go*
Roanoke	*The Computer Place*
Springfield	*Computer Workshop of North Virginia*
Virginia Beach	*Heathkit Electronic Center*
WASHINGTON	
Bellevue	*Altair Computer Center*
Kennewick	*C & J Electronics*
Longview	*Progress Electronics*
Pasco	*Riverview Electronics*
Richland	*C & J Electronics*
Seattle	*C-Com*
Seattle	*Empire Electronics*
Spokane	*Personal Computers*
WEST VIRGINIA	
Morgantown	*The Computer Corner*
Morgantown	*Electro Distributing Co.*
Ripley	*Thompson's Radio Shack*
CANADA	
Alberta (Calgary)	*The Computer Shop*
Ontario (Williowdale)	*Home Computer Centre*
Quebec (Montreal)	*Wang's Microcenter*
PANAMA	
Panama City	*Electrotecnia S.A.*
Panama City	*Sonitel, S.A.*
FRANCE	
Paris	*Computer Boutique*
SINGAPORE	*Inter-Trade (PTE) Ltd.*

SEE YOUR LOCAL Jim-pak electronic components DEALER TODAY!

For Dealer Information, write or phone JIM-PAK®, 1021 Howard Ave., San Carlos, California 94070 (415) 592-8097

Circle 201 on inquiry card.

HORIZON
THE COMPLETE COMPUTER

NorthStar ★ Computers
HORIZON

Look To The North Star HORIZON Computer.

HORIZON™— a complete, high-performance microprocessor system with integrated floppy disk memory. HORIZON is attractive, professionally engineered, and ideal for business, educational and personal applications.

To begin programming in extended BASIC, merely add a CRT or hard-copy terminal. HORIZON-1 includes a Z80A processor, 16K RAM, minifloppy™ disk and 12-slot S-100 motherboard with serial terminal interface — all standard equipment.

WHAT ABOUT PERFORMANCE?

The Z80A processor operates at 4MHZ — double the power of the 8080. And our 16K RAM board lets the Z80A *execute at full speed.* HORIZON can load or save a 10K byte disk program in less than 2 seconds. Each diskette can store 90K bytes.

AND SOFTWARE, TOO

HORIZON includes the North Star Disk Operating System and full extended BASIC on diskette ready at power-on. Our BASIC, now in widespread use, has everything desired in a BASIC, including sequential and random disk files, formatted output, a powerful line editor, strings, machine language CALL and more.

EXPAND YOUR HORIZON

Also available—Hardware floating point board (FPB); additional 16K memory boards with parity option. Add a second disk drive and you have HORIZON-2. Economical serial and parallel I/O ports may be installed on the motherboard. Many widely available S-100 bus peripheral boards can be added to HORIZON.

QUALITY AT THE RIGHT PRICE

HORIZON processor board, RAM, FPB and MICRO DISK SYSTEM can be bought separately for either Z80 or 8080 S-100 bus systems.

HORIZON-1 $1599 kit; $1899 assembled.
HORIZON-2 $1999 kit; $2349 assembled.

16K RAM—$399 kit; $459 assembled; Parity option $39 kit; $59 assembled. FPB $259 kit; $359 assembled. Z80 board $199 kit; $259 assembled. Prices subject to change. HORIZON offered in choice of wood or blue metal cover at no extra charge.

Write for free color catalogue or visit your local computer store.



2547 Ninth Street • Berkeley, California 94710 • (415) 549-0858

Circle 285 on inquiry card.